

# A CLASS OF FORMAL MODELS FOR LANGUAGES AND TRANSLATIONS

A Thesis Submitted  
In Partial Fulfilment of the Requirements  
for the Degree of  
**DOCTOR OF PHILOSOPHY**

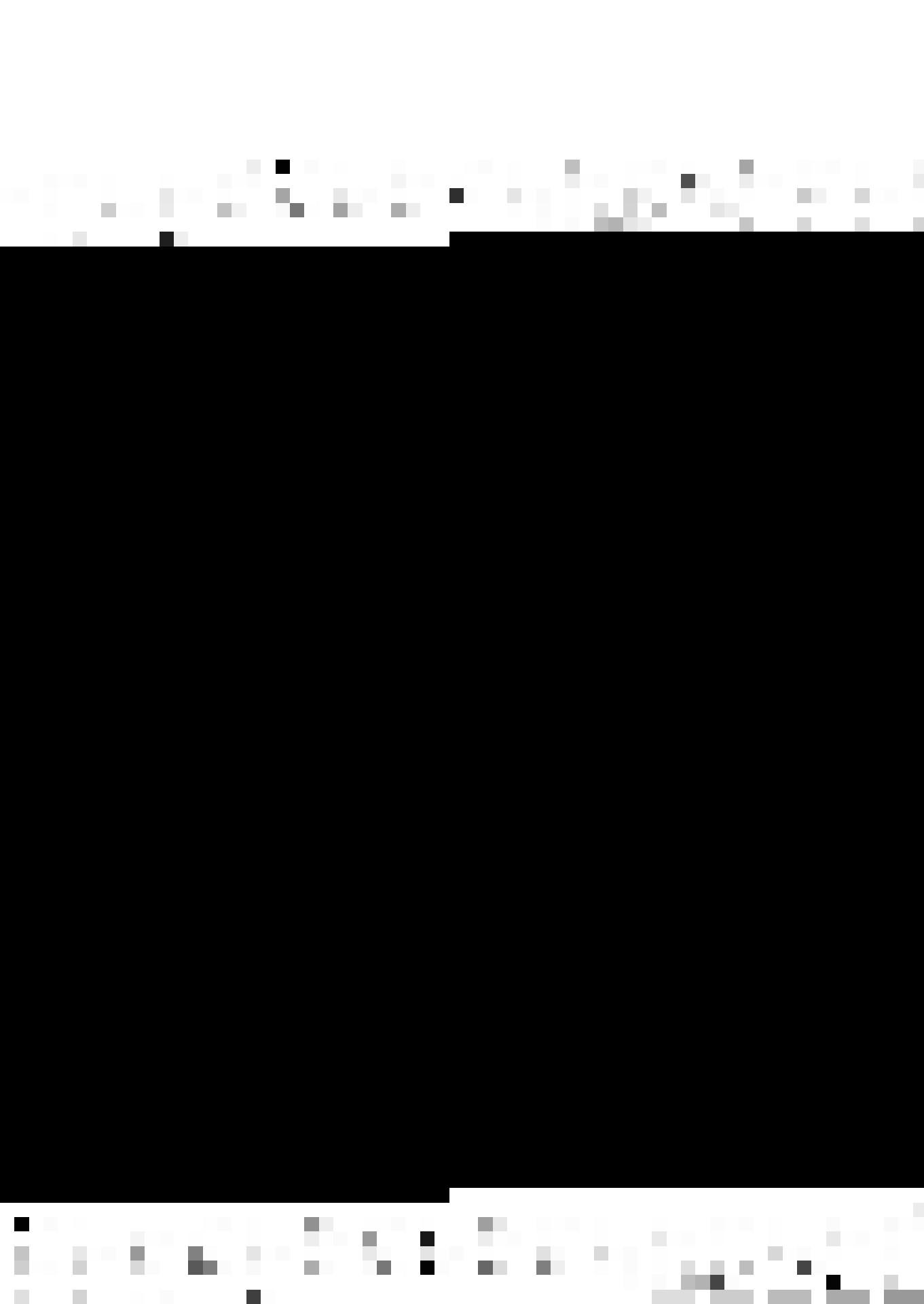
*by*  
**VIJAY KUMAR VAISHNAVI**

to the

**DEPARTMENT OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY KANPUR  
FEBRUARY, 1975**



to Dibba n



Thesis

001.642.5

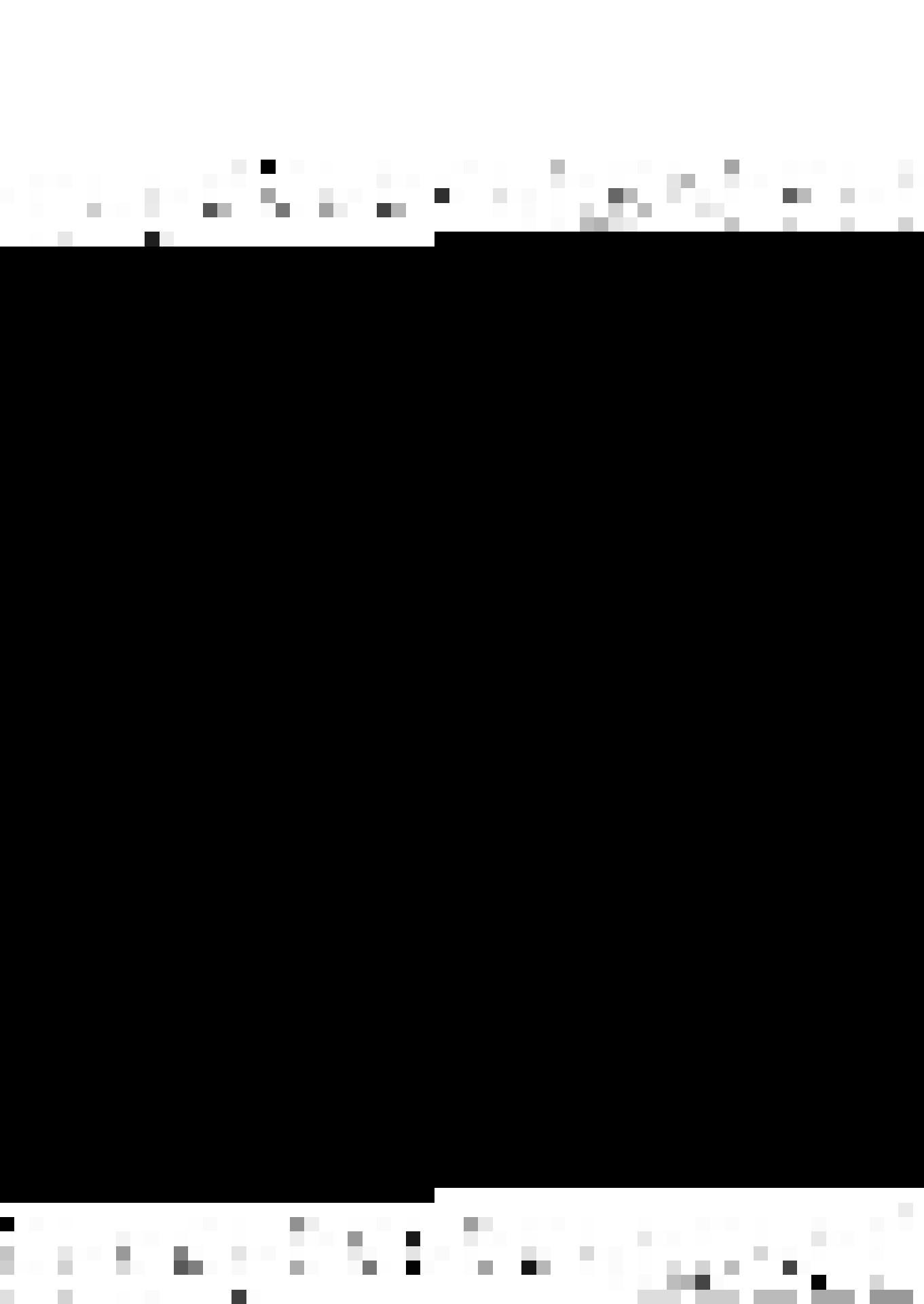
X 197



L.I.T. MANFUR  
CENTRAL LIBRARY  
Acc. No. A 45490

10 JAN 1976

EE-1975-D-VAL-CLA

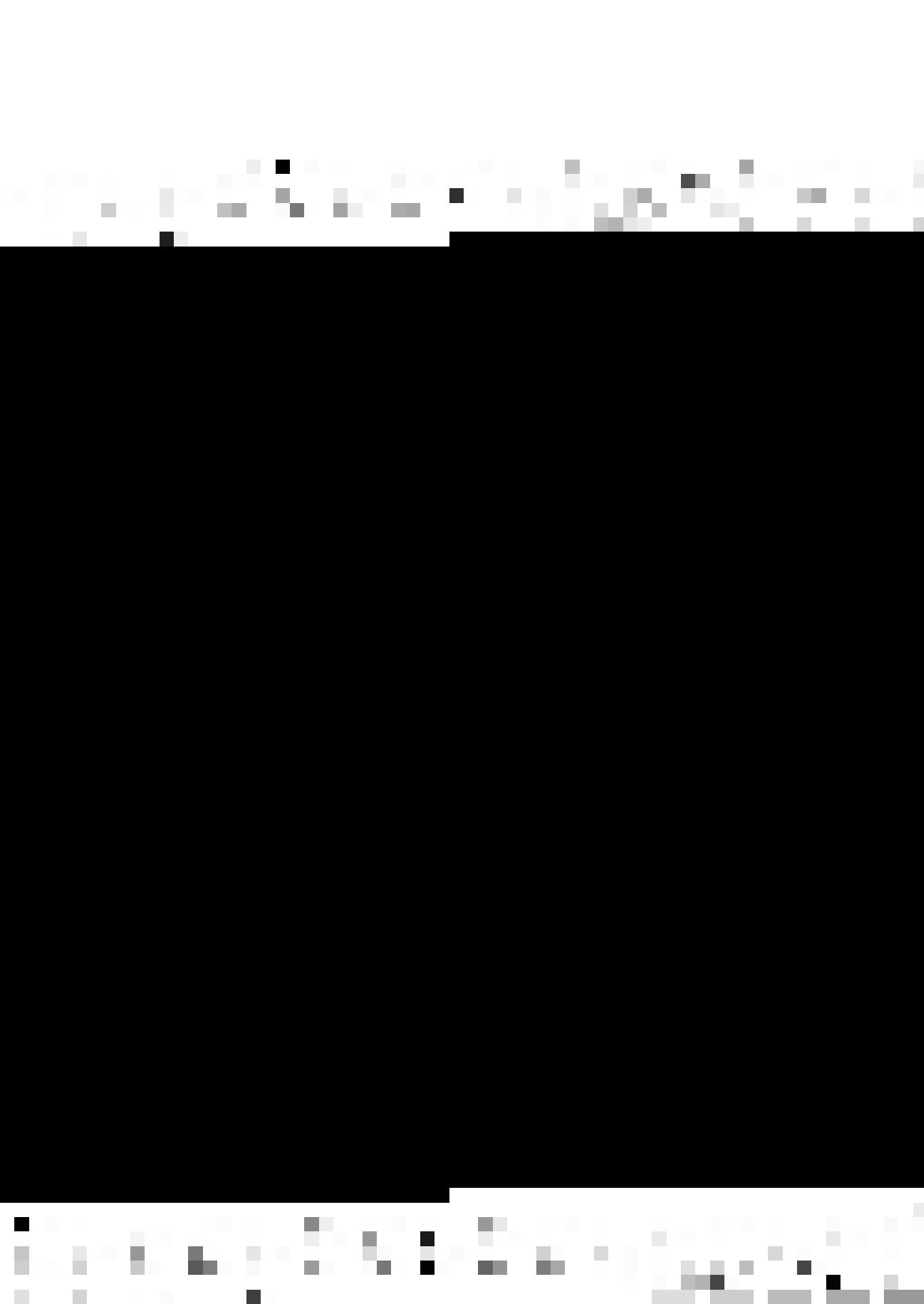


This is to certify that the work entitled " A CLASS OF FORMAL MODELS FOR LANGUAGE  
AND TRANSLATIONS " by V. K. Vaishnavi has been carried out under my  
supervision and has not been submitted elsewhere for a degree.

Date: January 6, 1975

  
(S. K. Basu)

1/6/75



## ACKNOWLEDGMENTS

I am deeply indebted to my thesis supervisor Dr. S.K. Basu for his expert guidance and encouragement at every stage of the work. He has found time and patience for many useful discussions. He has been a constant source of inspiration and encouragement.

I am grateful to Dr. H.V. Sahasrabuddhe for introducing me to the area of formal languages. I would like to express my thanks to Dr. V. Rajaraman and Dr. R. Narsimhan for their encouragement and interest in my work.

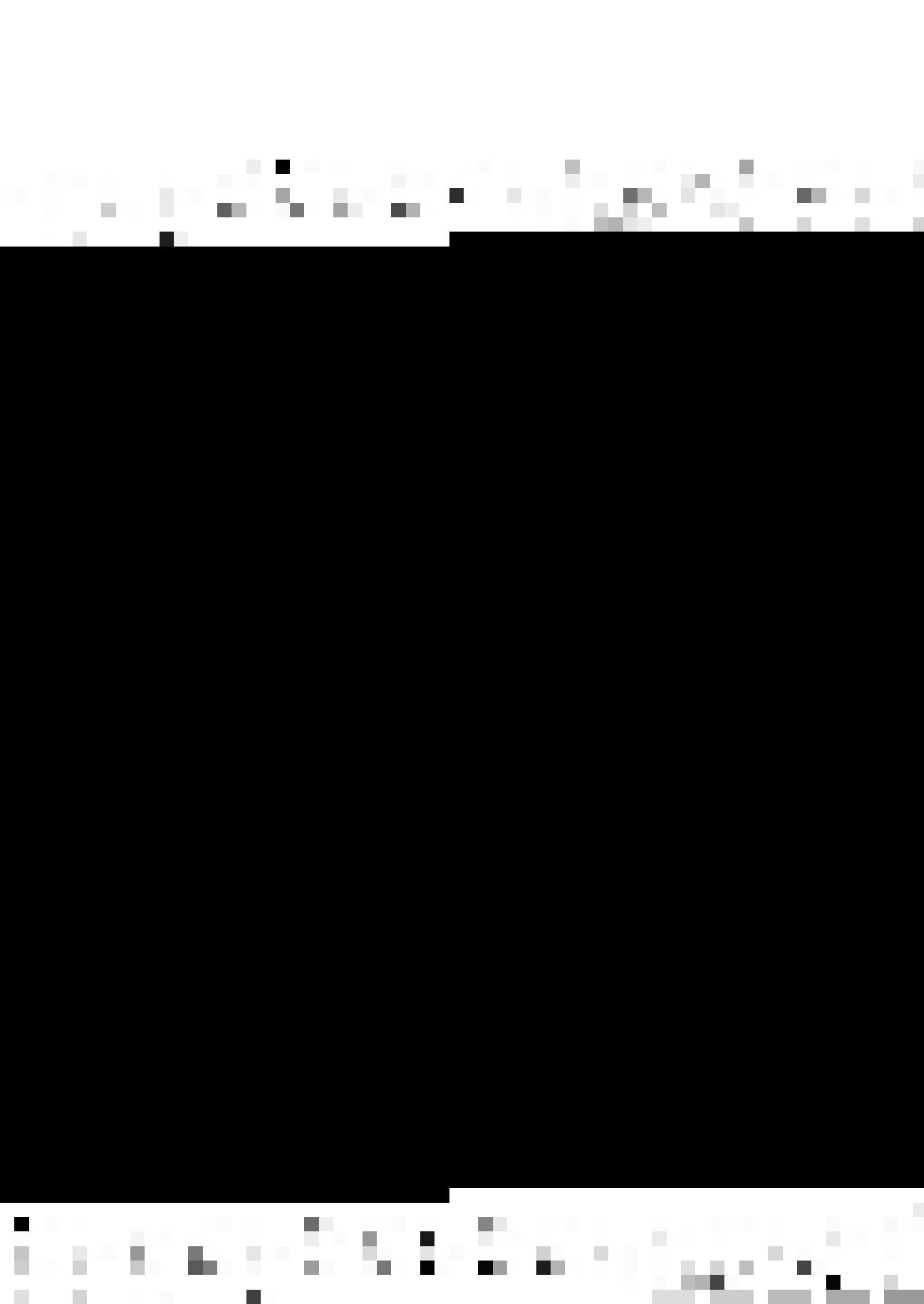
I am thankful to Mr. V.G. Kane for his timely help; particularly for going through an earlier draft in detail and offering many useful suggestions.

Finally, my acknowledgements are due to Mr. J.K. Misra for his excellent typing.

Vijay Kumar Vaishnavi

## CONTENTS

<u>Chapter</u>		<u>Page</u>
	SYNOPSIS	vi
	INTRODUCTION	1
1	BASIC DEFINITIONS AND RESULTS	7
	1.1 Coupled Grammars	7
	1.2 Coupled Translations and Coupled Languages	29
	1.3 n-Derivation Trees	39
	1.4 Simplification of Coupled Grammars	47
	1.5 Conclusions	56
2.	STRUCTURE RESULTS	59
	2.1 Basic Definitions and Results	59
	2.2 Structure Results - I	78
	2.3 Structure Results - II	94
	2.4 Structure Results - III	112
	2.5 Conclusions	121
3.	MACHINE CHARACTERIZATIONS	125
	3.1 n-Turn Generalized Checking Automata	125
	3.2 n-Generalized Checking Automata and n-Simple Coupled Languages	142
	3.3 n-Pushdown Assemblers	164
	3.4 Conclusions	172
4.	CLOSURE PROPERTIES AND DECISION PROBLEMS	175
	4.1 Closure Properties	175
	4.2 Decision Problems	189
	4.3 Conclusions	196



<u>Chapter</u>		<u>Page</u>
5.	RELATIONSHIP OF THE CLASS OF COUPLED LANGUAGES WITH SOME KNOWN CLASSES OF LANGUAGES	201
5.1	Context-Sensitive Languages and Coupled Languages	201
5.2	Simple Matrix Languages and Simple Coupled Languages	207
5.3	Equal Matrix Languages and Right Linear Coupled Languages	219
5.4	Conclusions	222
	DISCUSSION	223
	REFERENCES	226
Appendix	APPENDIX A	230
	APPENDIX B	231
	APPENDIX C	235
	APPENDIX D	238
	APPENDIX E	240
	APPENDIX F	242
	APPENDIX G	247
	APPENDIX H	262
	APPENDIX I	263
	APPENDIX J	270



SYNOPSIS

OF THE

Ph.D. DISSERTATION

on

A CLASS OF FORMAL MODELS FOR LANGUAGES  
AND TRANSLATIONS

by

VIJAY KUMAR VAISHNAVI

Department of Electrical Engineering  
Indian Institute of Technology  
Kanpur

There are two models in language theory which have been extensively studied and which have been particularly useful in specifying to certain extent the syntax and semantics of programming languages. These models are those of the context-free grammar and the syntax directed translation scheme. These models are not however, powerful enough for modelling accurately various structures and mappings that we come across in various areas such as programming languages and their compilation, natural language analysis and pattern analysis. As a result, recently there has been considerable interest in more powerful models of languages and translations which also lead to informative structural descriptions. It has also been felt that different formal models of languages (respectively,

translations) are of interest in language theory because a specific model that is capable of characterising certain aspects of a language structure (respectively, mappings between the structures of the domain and the range languages) naturally, may be unsatisfactory in some other respect. It is in this background that the work reported in this thesis should be viewed. Certain recently proposed models for languages such as the simple matrix languages and various generalizations of the syntax directed translation scheme can also be looked at from this angle.

We have in this thesis proposed and studied a class of generative grammars which we call 'coupled grammars', which can serve as models for languages as well as for translations. Context-free grammars and syntax directed translation schemes are special types of coupled grammars. Again, we have shown that the classes of simple matrix languages and equal matrix languages, which have been studied in recent years, are generated by certain restricted classes of coupled grammars. Thus the concept of a coupled grammar unifies and generalizes the concepts of a context-free grammar, an equal matrix grammar, a simple matrix grammar, and syntax directed translation scheme.

Intuitively, we can think of a coupled grammar of degree  $n \geq 1$ , as defining  $n$ -tuples of labeled rooted trees  $\langle T_1, \dots, T_n \rangle$ ,



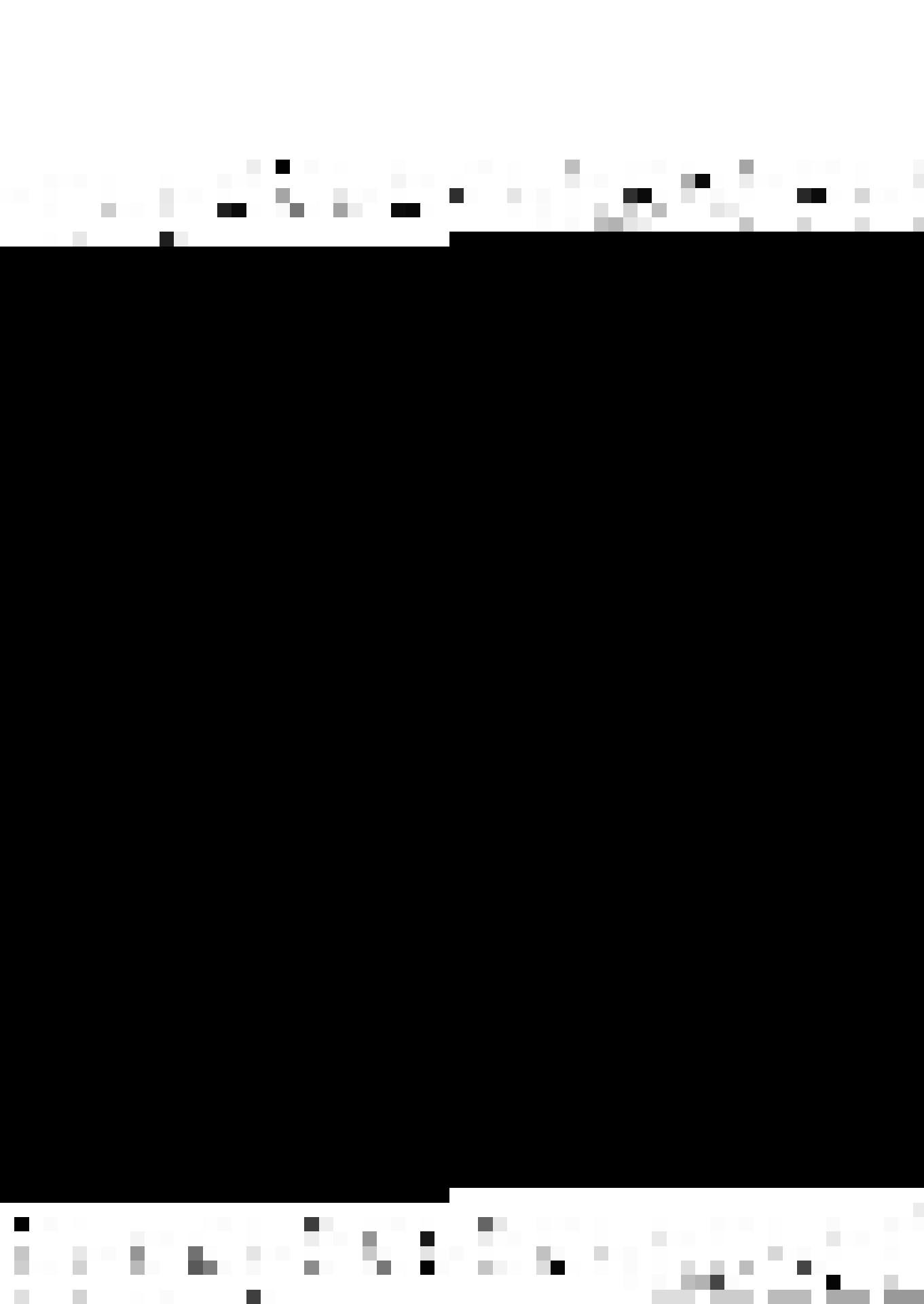
which we call  $n$ -derivation trees, where  $T_1$  is a derivation tree in a certain context-free grammar and each  $T_i$ ,  $2 \leq i \leq n$ , is formed from  $T_1$  by altering  $T_1$  at each node: deleting and/or introducing descendants with terminal labels, and reordering the nonterminal descendants according to some fixed rule. Within this framework of ideas, each sentence  $x$  of the language generated by a coupled grammar of degree  $n$ , called an  $n$ -coupled language, can be written as  $x_1 \dots x_n$  such that for a certain  $n$ -derivation tree  $\langle T_1, \dots, T_n \rangle$  defined by the grammar,  $x_i$  is the result of  $T_i$ . Similarly, the  $(m, n-m)$ -coupled translation,  $1 \leq m \leq n-1$ , generated by a coupled grammar  $G$  of degree  $n$  is  $\{(x, y) | x = x_1 \dots x_m, y = y_{m+1} \dots y_n\}$ , for an  $n$ -derivation tree  $\langle T_1, \dots, T_n \rangle$  defined by  $G$ ,  $x_i$  is the result of  $T_i$ ,  $1 \leq i \leq m$ , and  $y_i$  is the result of  $T_i$ ,  $m+1 \leq i \leq n\}$ . The concept of an  $n$ -derivation tree introduced informally above plays an important role in developing the theory, in this thesis.

The domain as well as the range of translations generated by coupled grammars are coupled languages. Thus a study of coupled languages is a contribution to the study of coupled translations, as well. Because of this fact and also because the class of coupled languages is interesting on its own right, as well, we primarily concentrate on coupled languages in the thesis.



We can divide the work reported in this thesis into four parts:

1. We identify certain complexity parameters of coupled grammars - degree, order and  $\beta$ -simplicity and impose certain restrictions on the productions. In this process, we define a number of subclasses of coupled languages and translations. Some of these are: n-coupled, n-simple coupled, n-right linear coupled, languages and  $(m,n)$ -coupled translations. We investigate the structures induced on languages and translations by restricting the productions and various associated complexity parameters. Some of the specific results are: the degree and order induce infinite proper hierarchies on the classes of coupled languages and translations.
2. We define a class of machines: n-turn generalized checking automata (generalization of n-turn checking automata) and show that these are machine characterizations of n-simple coupled languages. We also give the machine characterization of  $(1, n)$ -coupled translations.
3. We study some of the closure properties and decision problems of n-coupled and n-simple coupled, languages. We find that many of these results are similar to those of context-free languages.



4. Finally, we show that

- (i) the class of coupled languages contains the class of context-free languages and is properly contained in the class of context-sensitive languages,
- (ii) the class of  $n$ -simple coupled languages is the same as the class of simple matrix languages of degree  $n$ , and
- (iii) the class of  $n$ -right linear coupled languages is the same as the class of equal matrix languages of order  $n$ .



## INTRODUCTION

Lot of work has been done in the last two decades in the area of formal specification of languages and translations. The publication of the Algol Report (26) and the work of Irons (20) can be said to have led to most of the activity in this area.

The basic problem that this area is concerned with is the following (38): A given language or translation  $S$  is an infinite or a very large set and it has got a certain structure. We are interested in having a finite representation of  $S$  which also accurately represents its structure or certain aspects of its structure. In certain cases, we do not know exactly what structure  $S$  has got and we want the process of modelling it by devices which represent naturally various types of structures and the properties of these devices to help us in knowing the structure of  $S$  (modelling natural languages is a case in point).

It is this fact that we are not usually interested in just modelling a language or translation at the string level but at the structure level which has led to the study of a large variety of models.



There are two important types of models :

1. Generative grammars, and 2. Machines or automata. From the very early stages, a close relationship has been found between these two types of models and actually this relationship has contributed much to the theory that is available (see (18)).

Most of the work in formal generative grammars has been about four types of grammars: type 0 (or unrestricted rewriting systems which are equivalent in their generative power to Turing machines), type 1 (or context-sensitive grammars), type 2 (or context-free grammars) and type 3 (or regular grammars). This classification was given by Noam Chomsky (8). Out of the above types of grammars, context-free grammars have received the maximum attention (see (14)). Perhaps the reasons are that the structure that they represent can be expressed in a simple and neat manner through rooted labeled trees called derivation trees, and that they have been found to be quite useful in modelling aspects of programming languages. These grammars too, are not powerful enough in their generative power and capable of representing certain structures, as some applications demand; in fact, there does not exist a context-free grammar which can model Algol 60 (12). Type 0 grammars are least useful for representing the structure of a language. The



structure that context-sensitive grammars represent has not been found to be much informative. Regular grammars are too weak in their generative power.

Recently, there have been attempts to define and study new types of grammars which have their generative power in between those of context-free and context-sensitive grammars and have capability of representing different classes of structures. The motivation for it has been twofold. One is that in various applications like modelling programming languages, aspects of natural languages, and pattern analysis, we need models whose generative power is more than that of context-free grammars. Context-sensitive grammars are not a good solution to this problem because many of their important decision problems are unsolvable. The other motivation has been that we need different styles of grammars because each style represents faithfully certain types of structures and only awkwardly, if at all, certain other types of structures.

Need for different types of grammars which explicate different types of structures has been particularly felt while attempting to tackle the pattern analysis problem by the linguistic approach as against the decision theoretic approach adopted traditionally. The reason is that as against the case of natural languages where there has been a



search for a 'universal grammar' which will express the underlying structure of all natural languages (10), or the case of programming languages whose structure is not much varied, in pattern analysis we are obviously going to come across a variety of structures. Moreover, in pattern analysis the choice of the primitives as well as the equivalent of 'concatenation' is open. Also, it may be noted that even in case of natural languages, for practical applications we will be interested in grammars which bring out certain structures of the languages, rather than in completely adequate grammars.

The work on the models: matrix grammars (1), simple matrix grammars (19), scattered context grammars (17), random context grammars (40), time varying grammars (32), state grammars (21), programmed grammars (31), indexed grammars (2), etc., is an attempt in this direction. The corresponding work in translation models has been certain generalizations of syntax-directed translation schemes (20,25,3). Among these are the works of Petrone (29), Aho and Ullman (5), Knuth (23) and Vaishnavi (37). The work reported in this thesis also, is an attempt in this direction.

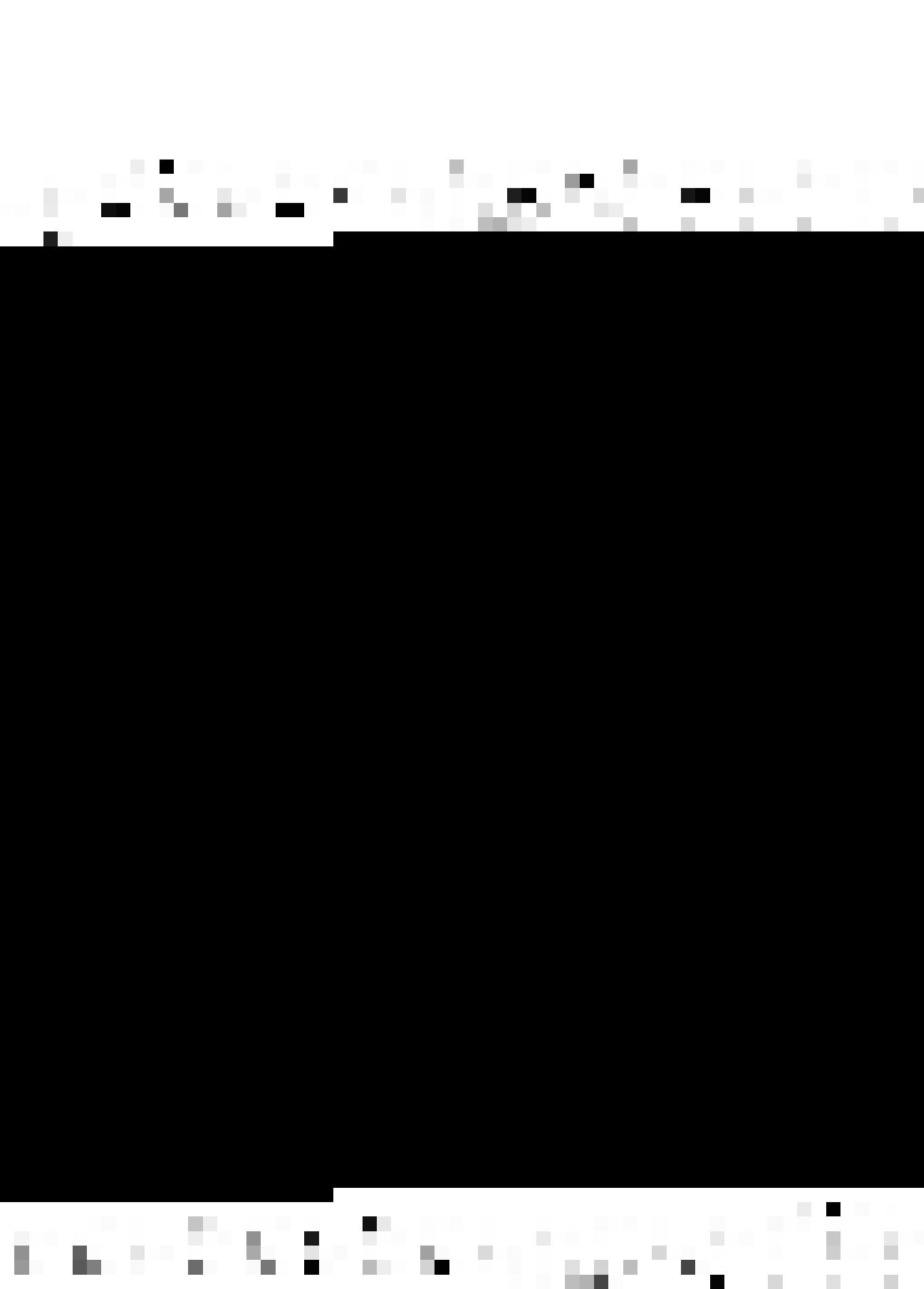
In this thesis, we define and study a class of formal grammars which we call 'coupled grammars'. These grammars can serve as formal models for languages as well as for



translations. As translation models, these have been partly studied in Vaishnavi (37). Coupled grammars are generalizations of syntax directed translation schemes and can also be looked at as natural extensions of context-free grammars. Context-free grammars and syntax directed translation schemes are, in fact, special types of coupled grammars. Again, we have shown that the classes of simple matrix languages (19) and equal matrix languages (34) which have been studied in recent years, are generated exactly by classes of certain restricted types of coupled grammars. Thus the concept of a coupled grammar unifies and generalizes the concepts of a context-free grammar, an equal matrix grammar, a simple matrix grammar, and a syntax directed translation scheme.

The domain as well as the range of translations generated by coupled grammars are coupled languages. Thus a study of coupled languages is a contribution to the study of coupled translations as well. Because of this fact and also because the class of coupled languages is interesting on its own right, as well, we primarily concentrate in the thesis on coupled languages.

In Chapter 1, we give the basic definitions and results required for the rest of the thesis.



In Chapter 2, we investigate how the various complexity parameters of coupled grammars: degree,  $\mu$ -simplicity and order, induce structure on the languages and translations generated by them and how these structures are related with each other.

In Chapter 3, we give the machine characterizations for a subclass of coupled languages called simple coupled languages of degree  $n$  and a subclass of coupled translations called  $(1, n)$ -coupled translations.

In Chapter 4, we investigate some of the closure properties and decision problems of coupled languages.

In Chapter 5, we discuss the relationship between the class of coupled languages and some known classes of languages viz. the classes of context-sensitive languages, simple matrix languages, and equal matrix languages.

Finally, we conclude the thesis with a discussion on the work reported in the thesis and the work that can be done further.



## CHAPTER 1

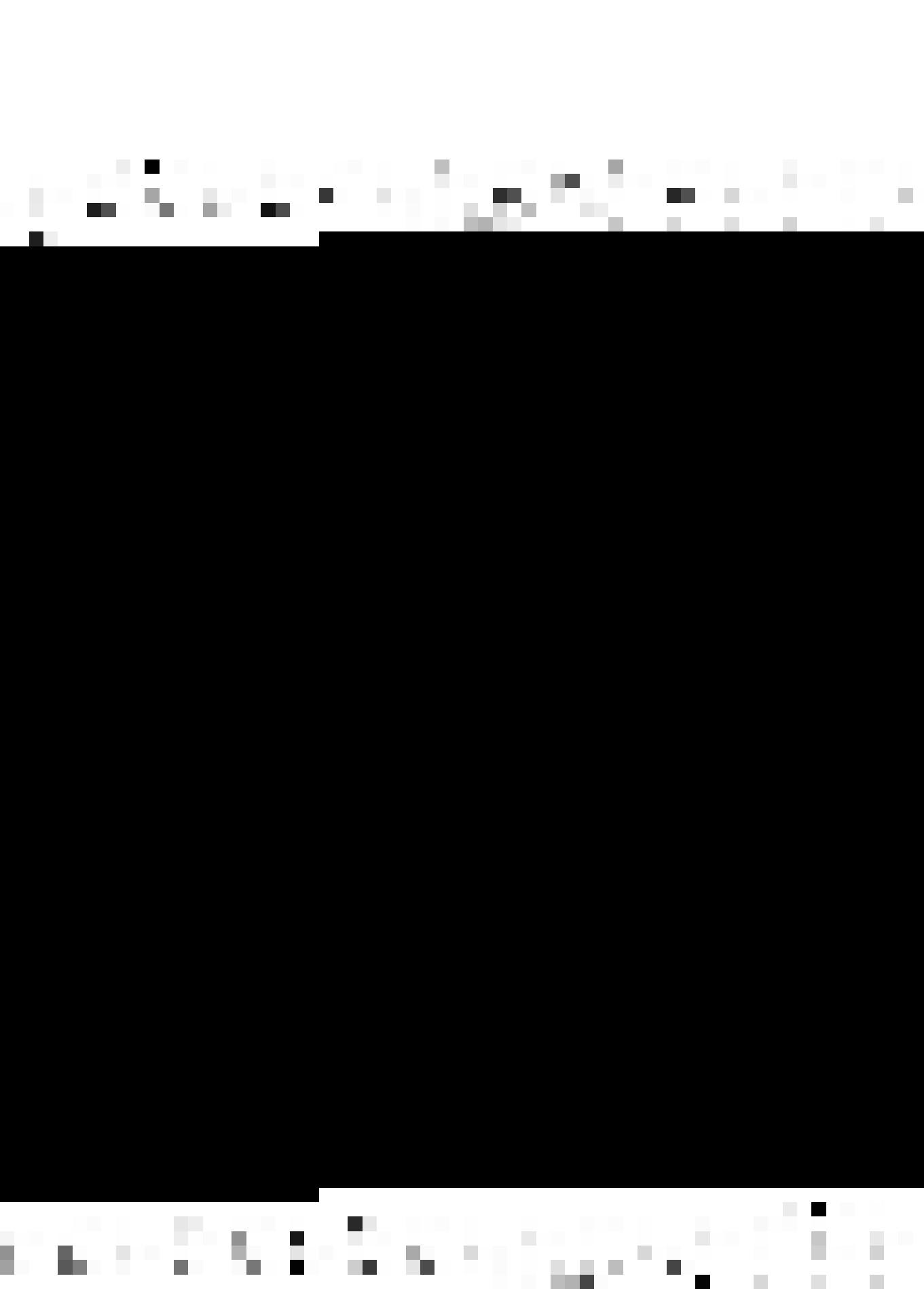
### BASIC DEFINITIONS AND RESULTS

This chapter lays the foundation of the later chapters in terms of the basic definitions and results.

In section 1.1, we give various definitions and a result concerning coupled grammars. In section 1.2, we define coupled translations, coupled languages and related concepts. In section 1.3, we introduce the concept of  $n$ -derivation trees,  $n \geq 1$ , which is similar to the concept of derivation trees in context-free grammars, and plays an important role in studying the structure of coupled languages. In section 1.4, we investigate some of the possible simplifications of coupled grammars. Finally, in section 1.5, we give some conclusions.

#### 1.1 COUPLED GRAMMARS:

Coupled grammars are generalizations of syntax directed translation schemes( 3 ). These can also be looked at as natural extensions of context-free grammars. Again, syntax directed translation schemes are generalizations of context-free grammars. In order to appreciate all this, we shall



first review the definitions of context-free grammars, syntax directed translation schemes, and related concepts before formally defining coupled grammars.

A context-free grammar  $G$  is defined as:

$$G = (V_N, V_T, P, S),$$

where  $V_N$  and  $V_T$  are disjoint finite sets of nonterminals and terminals, respectively.  $S$ , in  $V_N$ , is the start symbol.  $P$  is a finite set of productions of the type  $A \rightarrow \alpha$ , where  $A$  is in  $V_N$  and  $\alpha$  is in  $(V_N \cup V_T)^*$ . If  $A \rightarrow \alpha$  is in  $P$ ,  $\beta$  and  $\gamma$  in  $(V_N \cup V_T)^*$ , then we write  $\beta A \xrightarrow[G]{} \beta\alpha$ . The relation  $\xrightarrow[G]^*$  is the reflexive-transitive closure of  $\xrightarrow[G]$ .

The language generated by  $G$ , called a context-free language is:

$$\{w | w \text{ is in } V_T^* \text{ and } S \xrightarrow[G]^* w\}.$$

\*\*\*

One of the weaknesses of context-free grammars is the lack of 'cross-referencing' between nonterminals in the process of the derivation of a sentence. This is the reason why the language  $L = \{a^i b^i c^i | i \geq 1\}$  is not a context-free language(14). This weakness of context-free grammars makes them unsuitable for many applications. The context-sensitive grammars, on the other hand, are too powerful in this respect. The context-sensitive grammars don't, however, have many of the nice properties of context-free grammars; also they don't lead



to very informative structural descriptions. This is the motivation that we have for the definition of coupled grammars. In these grammars we can have this 'cross-referencing' between nonterminals in a controlled manner.

A syntax directed translation scheme  $H$  is defined(3) as:

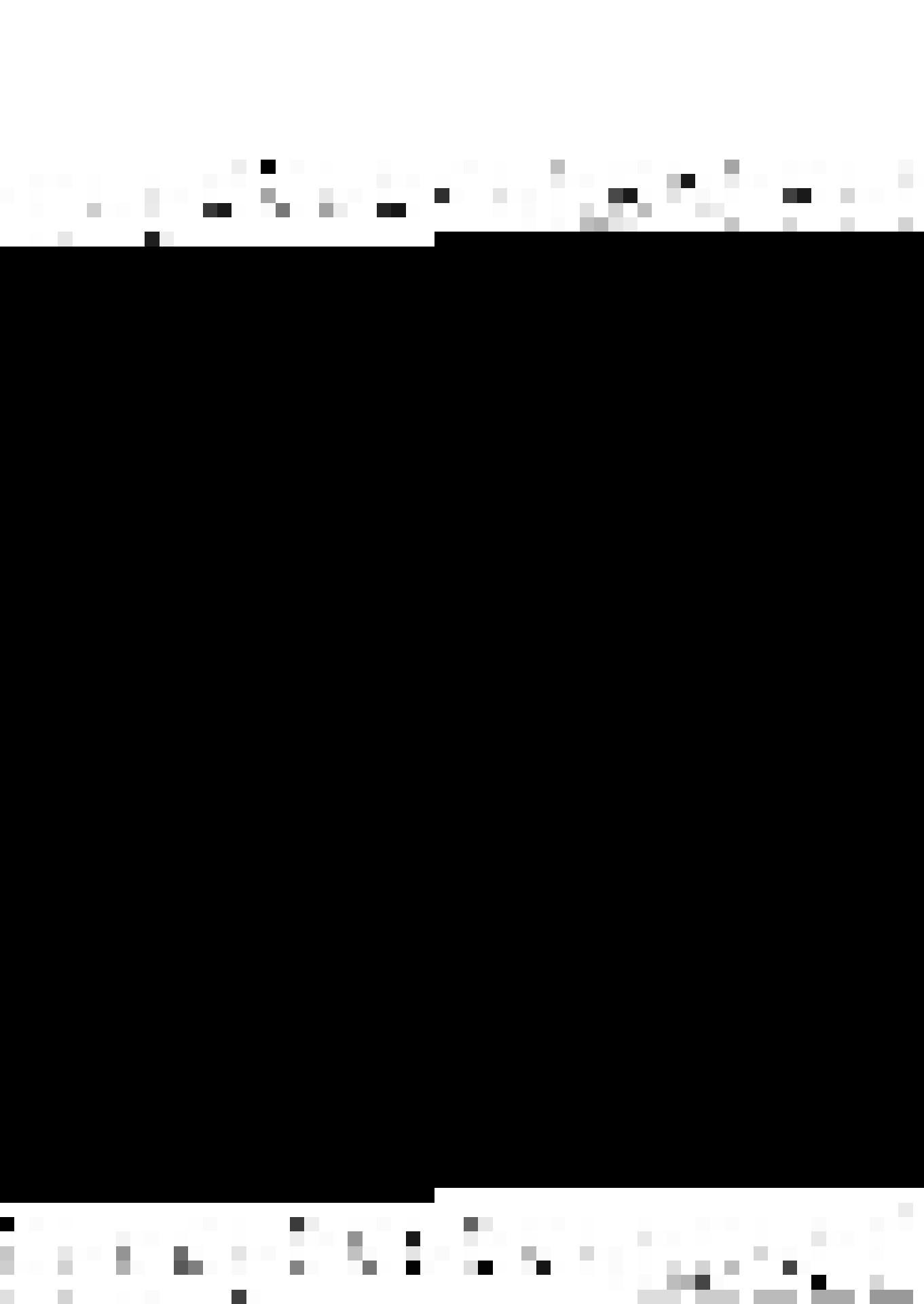
$$H = (V_N, V_{T_1}, V_{T_2}, R, S),$$

where  $V_N$ ,  $V_{T_1}$  and  $V_{T_2}$  are finite sets of nonterminals, input symbols and output symbols, respectively.  $V_N$  is disjoint from  $V_{T_1} \cup V_{T_2}$ .  $S$ , in  $V_N$ , is the start symbol.  $R$  is a finite set of rules, a term which we shall define fully after giving the following auxiliary definition:

A translation form of  $H$  is a triple  $(\alpha, \beta, \pi)$ , where  $\alpha$  is in  $(V_N \cup V_{T_1})^*$ ,  $\beta$  is in  $(V_N \cup V_{T_2})^*$  and  $\pi$  is a permutation<sup>1</sup>. The number of occurrences of nonterminals in  $\alpha$  and  $\beta$  must be equal, say  $k$  in each, and the number of appearances of each nonterminal in  $\alpha$  and  $\beta$  should be the same.  $\pi$  must then be a permutation on  $k$  objects. For all  $i$ ,  $1 \leq i \leq k$ , the  $i$ -th nonterminal of  $\alpha$  (from the left) is the same as the

---

1. A permutation  $\pi$  on  $k$  objects will always be denoted by  $[i_1, \dots, i_k]$ , where  $\{i_j | 1 \leq j \leq k\}$ , is a set of distinct integers between 1 and  $k$ .  $\pi(j)$  is defined to be  $i_j$ .  $\hat{\pi}(j)$  is that  $s$  such that  $\pi(s) = j$ .



$\pi(i)$ -th nonterminal of  $\beta$  (from the left). We say that the  $i$ -th nonterminal of  $\alpha$  and the  $\pi(i)$ -th nonterminal of  $\beta$  correspond.

A rule is an object  $A \rightarrow (\alpha, \beta, \pi)$ , where  $A$  is in  $V_N$  and  $(\alpha, \beta, \pi)$  is a translation form.

Suppose  $(\alpha_1, \beta_1, \pi_1)$  is a translation form of  $H$  and  $A$  the  $i$ -th nonterminal of  $\alpha_1$ , in order from the left. Also, suppose  $A \rightarrow (\gamma, \delta, \pi)$  is a rule. Then we can construct a translation form  $(\alpha_2, \beta_2, \pi_2)$  from  $(\alpha_1, \beta_1, \pi_1)$  by replacing the  $i$ -th nonterminal of  $\alpha_1$  by  $\gamma$  to obtain  $\alpha_2$  and the  $\pi_1(i)$ -th nonterminal of  $\beta_1$  by  $\delta$  to obtain  $\beta_2$ .  $\pi_2$  is the permutation such that the nonterminals of  $\alpha_1$  other than the  $i$ -th correspond to the same symbol in  $\beta_2$  as in  $\beta_1$  and each nonterminal of  $\gamma$  corresponds to the nonterminal of  $\delta$  to which it corresponded according to  $\pi$ . We then write  $(\alpha_1, \beta_1, \pi_1) \xrightarrow{H} (\alpha_2, \beta_2, \pi_2)$ .  $\xrightarrow{H}$  is the reflexive-transitive closure of  $\Rightarrow$ .

The syntax directed translation defined by  $H$  is:

$$\left\{ (x, y) | (S, S) \xrightarrow{H}^* (x, y) \text{ and } x \text{ is in } V_{T_1}^*, y \text{ is in } V_{T_2}^* \right\}^1. \quad ***$$

Observe the role of 'correspondence' between nonterminals in the derivation of a translation element  $(x, y)$  by a syntax directed translation scheme. There is a controlled cross-

- 
1. Note that permutations have been omitted from the translation forms related by  $\xrightarrow{H}^*$  since no information is lost in doing so.



referencing between nonterminals. It is this idea that we extend and adapt to the definition of coupled grammars as translation as well as language generating devices. To illustrate the latter point, we observe that we can define the

'language generated' by a syntax directed translation scheme

$H$  as  $\{w \mid (S, S) \xrightarrow[H]{*} (x, y), x \text{ is in } V_{T_1}^*, y \text{ is in } V_{T_2}^* \text{ and } w = xy\}$ .

According to our terminology a syntax directed translation scheme is essentially a coupled grammar of degree 2 while a context-free grammar is a coupled grammar of degree one. We actually define coupled grammars of degree  $n$ ,  $n \geq 1$ . To get an idea of the motivation for defining coupled grammars of

degree higher than 2, we claim that the language

$\{a_1^{i_1}a_2^{i_2}a_3^{i_3}a_4^{i_4}a_5^{i_5} \mid i_j \geq 1\}$  cannot be generated by any coupled grammar of degree 2 (See Lemma 2.2.1). Actually the degree of coupled grammars induces an infinite proper hierarchy on the languages and translations generated by these grammars (See section 2.2).

Now, let us give the formal definition of coupled grammars:

Let  $n \geq 1$ . A coupled grammar of degree n, also called an  $n$ -coupled grammar ( $nCG$ ) is a 4-tuple:

$$G = (V_N, V_T, P, (S)^n)^1, \text{ where}$$

1. For any  $X$ ,  $(X)^n$  will always denote the  $n$ -tuple  $(X, \dots, X)$ . Note that  $X^n$  will denote  $X \dots X$ ; a string which comprises of  $n$   $X$ 's.



1.  $V_N$  and  $V_T$  are finite disjoint sets of nonterminals and terminals, respectively.
  2.  $V_T = V_{T_1} \cup \dots \cup V_{T_n}$  and  $V_{T_i}$  is the set of i-terminals,  $1 \leq i \leq n$ ;
  3.  $(S)^n$  is the start form, where  $S$  is in  $V_N$  and is called the start symbol;
  4.  $P$  is a finite set of productions, like  $A \rightarrow \alpha$ , where  $A$  is in  $V_N$  and  $\alpha$  is a 'form' in  $G$  which is defined as follows:  
 A form in  $G$  is an  $n$ -tuple:  $((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$ <sup>†</sup>, where  $\alpha_i$ ,  $1 \leq i \leq n$ <sup>‡</sup>, is in  $(V_N \cup V_{T_i})^*$ , the number  $k$  of occurrences of nonterminals in  $\alpha_1$  equals the number of occurrences of nonterminals in  $\alpha_j$ ,  $2 \leq j \leq n$ , and  $\pi_i$ ,  $1 \leq i \leq n$ , is a permutation on  $k$  objects. The  $\pi_j(i)$ -th nonterminal in  $\alpha_1$ ,  $1 \leq i \leq k$ , in order from the left, must be the same as the  $\pi_j(i)$ -th nonterminal in  $\alpha_j$ ,  $2 \leq j \leq n$ , in order from the left. For each  $i$ ,  $1 \leq i \leq k$ , the  $\pi_1(i)$ -th nonterminal in  $\alpha_1, \dots, \pi_n(i)$ -th nonterminal in  $\alpha_n$  (all in order from the left) are said to correspond with each other in the form. If  $k = 0$ , then  $\pi_i$ ,  $1 \leq i \leq n$ , is null by convention and the form is called a terminating form. Those productions whose right hand sides
1. If some permutation  $\pi_j$ ,  $1 \leq j \leq n$ , is omitted from the form, it will always be understood to be the identity permutation.
  2. From now onwards we shall usually omit the bounds on a parameter (like  $i$ , here) if these are obvious from the context.



are terminating forms are called terminating productions while others are called nonterminating productions.

$G$  is a coupled grammar (CG) iff it is an nCG for some  $n \geq 1$ .

### Example 1.1.1:

Let us give an example of a 3-coupled grammar:

$G = (V_N, V_T, P, (3)^3)$ , where  $V_N = \{S, A, B\}$ ,  $V_{T_1} = \{a, b\}$ ,  $V_{T_2} = \{a\}$ , and  $V_{T_3} = \{b\}$ , so that  $V_T = \{a, b\}$ , and  $P$  comprises of the productions:

1.  $S \rightarrow ((AB, [2,1]), TA, (AB, [2,1]))$ ,
2.  $A \rightarrow ((AaBAb, [2,3,1]), (aaABA, [2,1,3]),$   
 $(AAbbB, [3,2,1]))$ ,
3.  $A \rightarrow (a, a, \epsilon)$ , and
4.  $B \rightarrow (b, \epsilon, \epsilon)$ ,

is a 3CG.

The right hand sides<sup>1</sup> of the productions in  $P$  are forms in  $G$ . The productions 1 and 2 are nonterminating productions while 3 and 4 are terminating productions. In the right hand side of the production 2, the second, third, and first nonterminals of  $AaBAb$  correspond, respectively, with the second, first, and third nonterminals of  $aaABA$ , and with the third, second, and first nonterminals of  $AAbbB$  (all in order from the



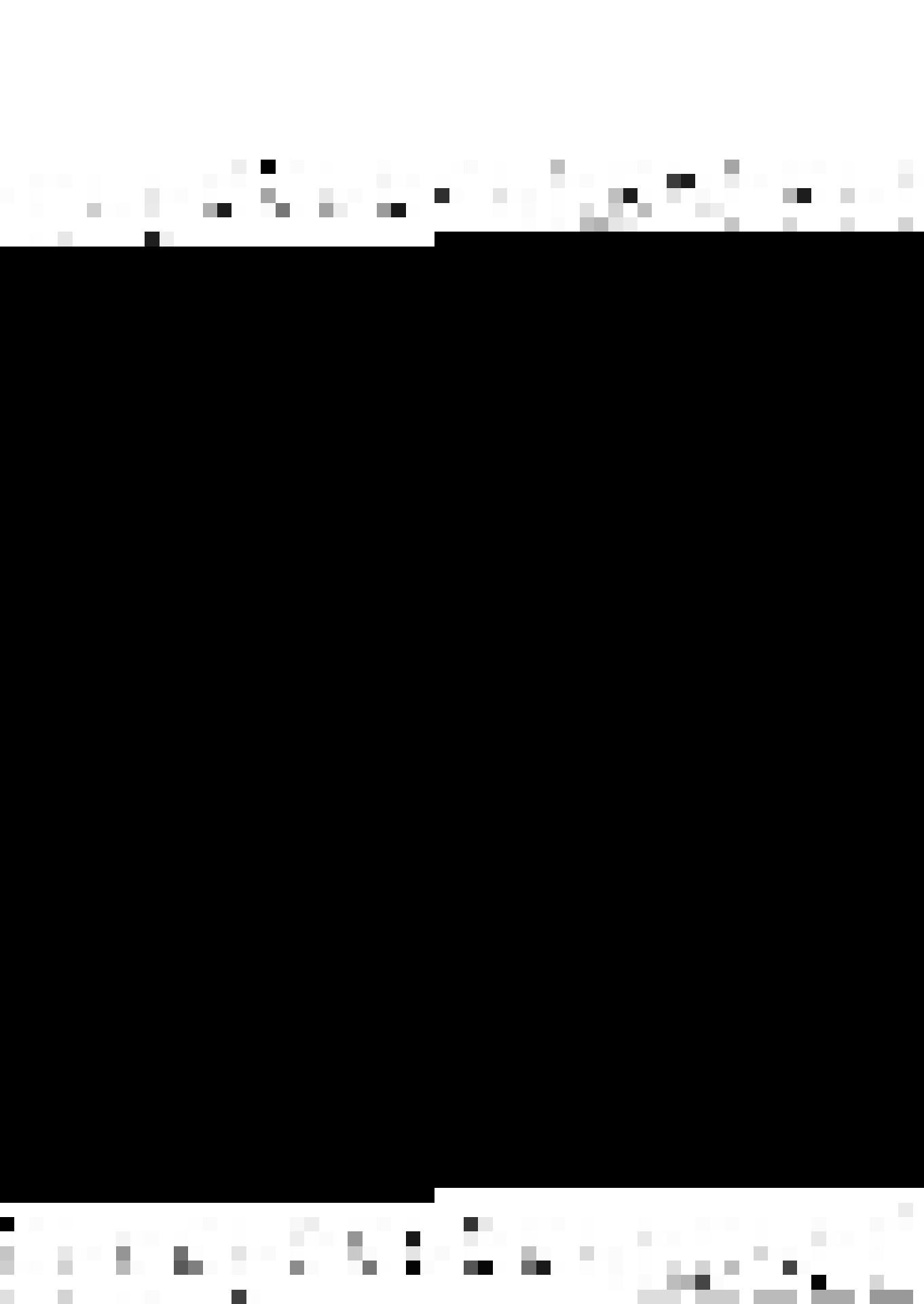
We note that nothing is gained by having the permutation in the first component of a form as a non-identity permutation; having it as the identity permutation makes the productions look less complex. We will come back to this point later.

In order that n-coupled grammars<sup>1</sup> may serve as generative devices for translations and languages, we will have to define what is meant by a derivation in these grammars. Let us proceed with doing this.

Let  $G = (V_N, V_T, P, (S)^n)$  be an nCG and let  $\alpha_1 = ((\alpha_{11}, \gamma_{11}), \dots, (\alpha_{1n}, \gamma_{1n}))$  and  $\alpha_2 = ((\alpha_{21}, \gamma_{21}), \dots, (\alpha_{2n}, \gamma_{2n}))$ , be two forms in it. Let some nonterminal, say,  $A$  be the  $\gamma_{11}(i)$ -th nonterminal in  $\alpha_{11}$  for some  $i$  (in order from the left) and let there be some production  $\psi: A \rightarrow ((\gamma_1, \gamma_1), \dots, (\gamma_n, \gamma_n))$ , in  $P$ . Let there be  $m$  nonterminals in  $\gamma_1$  (and hence in each  $\gamma_i$ ,  $1 \leq i \leq n$ ).

$\alpha_1$  is said to directly derive  $\alpha_2$  in  $G$ , symbolically written as<sup>2</sup>  $\alpha_1 \xrightarrow{G} \alpha_2$ ; (using production  $\psi$  in 'expanding'  $A$ , the  $\gamma_{11}(i)$ -th nonterminal of  $\alpha_{11}$ , in order from the left) iff the following holds:

- 
1. Throughout, we will follow the convention that if the value of a certain parameter is not specified, the parameter can have any permissible value. Thus, here  $n \geq 1$ .
  2. 'G' may be dropped whenever no ambiguity is caused by doing so.



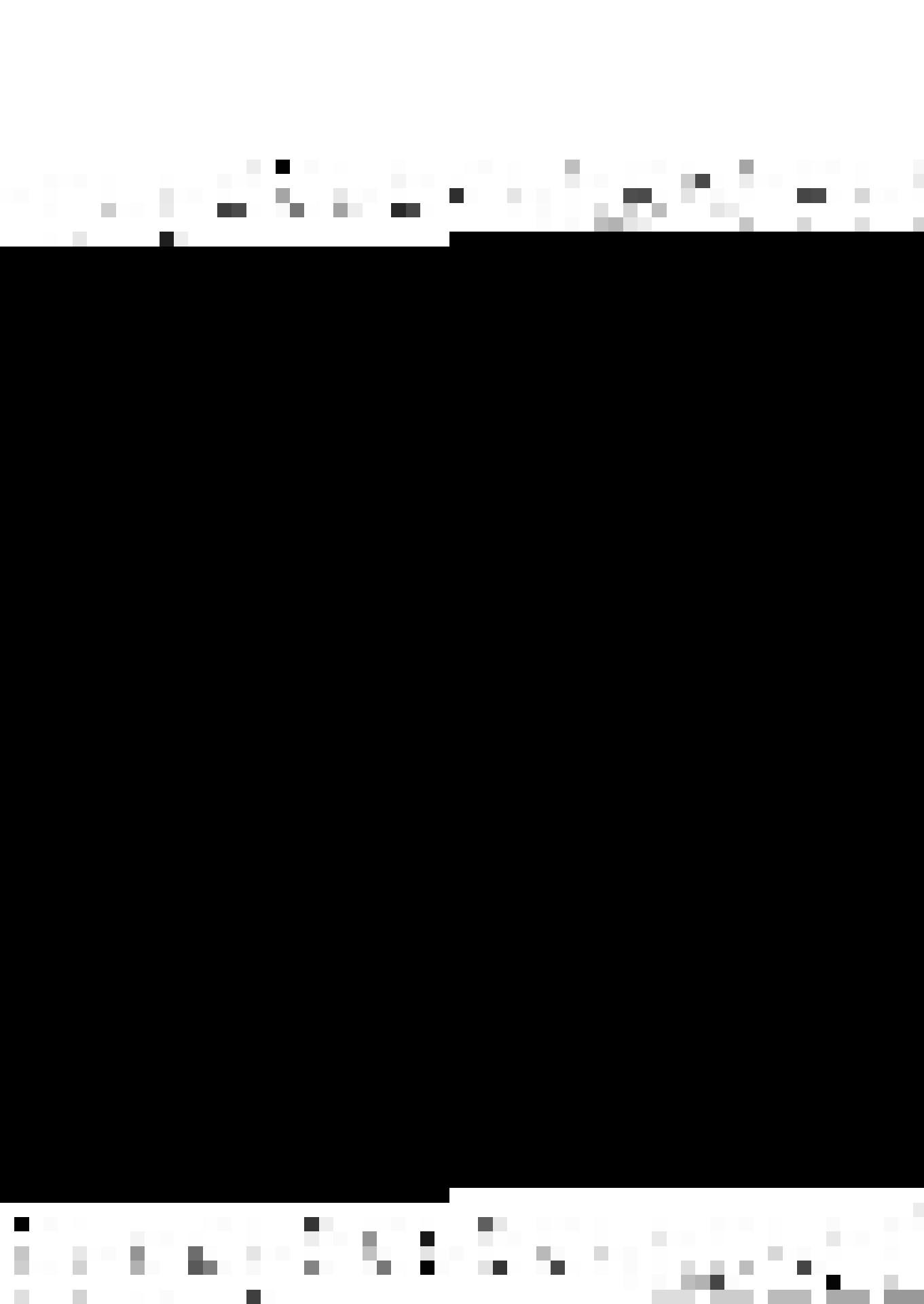
1.  $\alpha_{21}$  and  $\alpha_{2j}$ ,  $2 \leq j \leq n$ , are obtained by replacing the  $\pi_{1j}(i)$ -th and the corresponding nonterminal in  $\alpha_{1j}$  by  $\gamma_1$  and  $\gamma_j$ , respectively.

2.  $\pi_{2j}$ 's, are obtained from  $\pi_{1j}$ 's and  $\pi_j$ 's such that the nonterminals in  $\alpha_2$  which are not the result of expansion, have got the same 'correspondence' as they had in  $\alpha_1$ , and the nonterminals which are the result of expansion, have got the same correspondence as they had in the right hand side of the production  $\beta$ . Formally, for all  $j$ ,  $1 \leq j \leq n$ , and (a) for all  $k < i$ , if  $\pi_{1j}(k) < \pi_{1j}(i)$ , then  $\pi_{2j}(k) = \pi_{1j}(k)$ , and if  $\pi_{1j}(k) > \pi_{1j}(i)$ , then  $\pi_{2j}(k) = \pi_{1j}(k) + m - 1$ , (b) for all  $k > i$ , if  $\pi_{1j}(k) < \pi_{1j}(i)$ , then  $\pi_{2j}(k + m - 1) = \pi_{1j}(k)$ , and if  $\pi_{1j}(k) > \pi_{1j}(i)$ , then  $\pi_{2j}(k + m - 1) = \pi_{1j}(k) + m - 1$ , and (c) for all  $d$ ,  $1 \leq d \leq m$ ,  $\pi_{2j}(i+d-1) = \pi_{1j}(i) + \pi_j(d) - 1$ .

We say that a form  $\alpha$  (in  $G$ ) derives a form  $\beta$  (in  $G$ ), symbolically written as<sup>1</sup>  $\underset{G}{\alpha \xrightarrow{*} \beta}$ , iff there are forms  $\alpha_0, \dots, \alpha_k$ , in  $G$ ,  $k \geq 0$ , such that  $\alpha_0 = \alpha$ ,  $\alpha_k = \beta$  and  $\alpha_i \Rightarrow \alpha_{i+1}$ ,  $0 \leq i < k$ .

$(S)^n \xrightarrow{*} (w_1, \dots, w_n)$ , is said to be a derivation of  $w$  in  $G$  if  $w_i$ ,  $1 \leq i \leq n$ , is in  $V_{T_i}^*$  and  $w = w_1 \dots w_n$ .

- Again, 'G' may be dropped, if no ambiguity is caused by doing so.



Example 1.1.2:

Let us illustrate the concept of one form directly deriving another form in an n-coupled grammar. Let  $G$  be the grammar given in Example 1.1.1, then

$$(AB, (BA, [2,1]), AB) \xrightarrow{G} (AaBabB, (BaaABA, [4,3,2,1])), \\ (AabbBB, [1,3,2,4]))$$

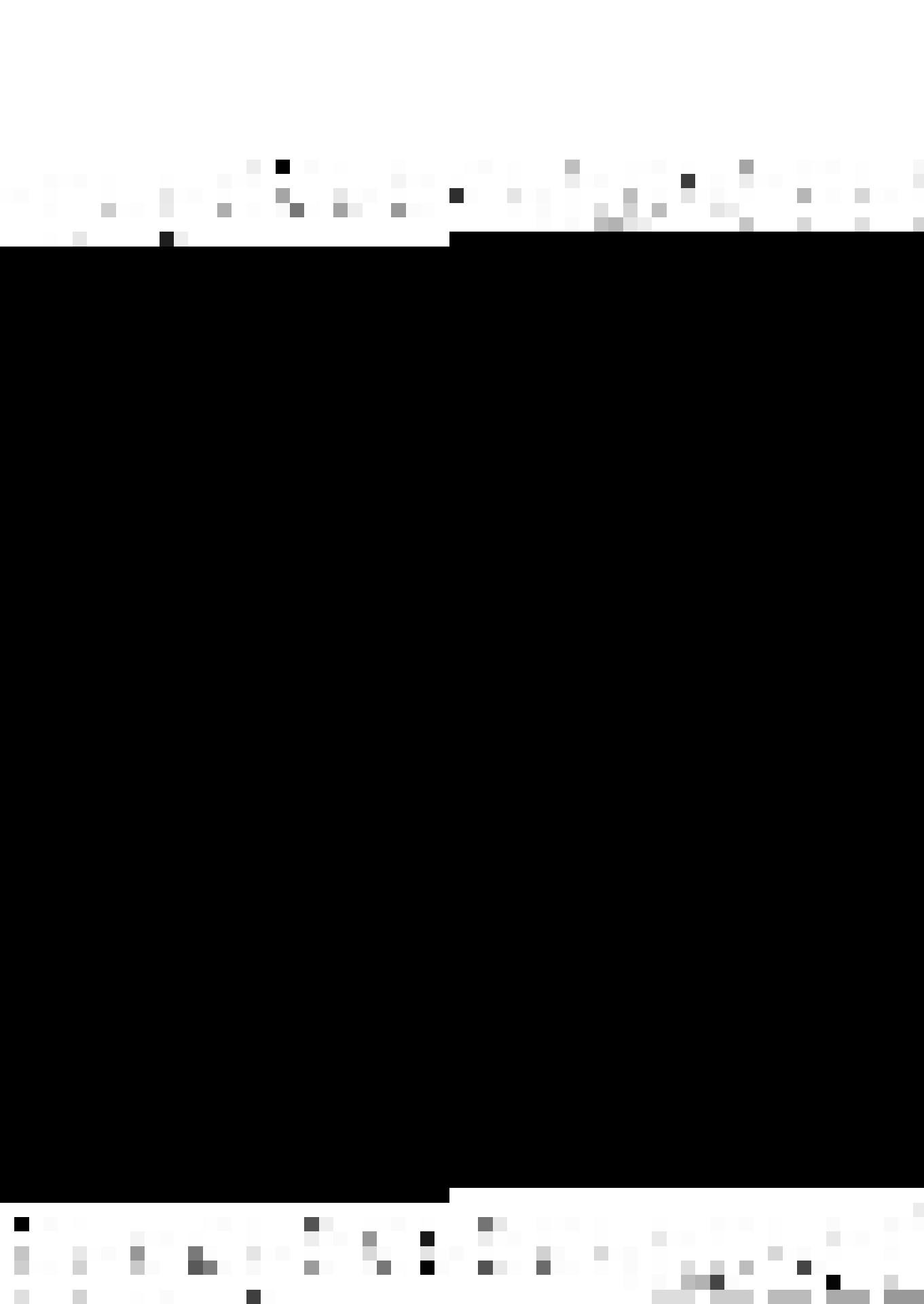
using the production 2 in expanding the nonterminal A.

\*\*\*

It should be clear at this point how we will be defining<sup>1</sup> the languages and translations generated by n-coupled grammars. For an n-coupled grammar  $G = (V_N, V_T, P, (S)^n)$ , we will define the language generated by  $G$  (which we shall call an n-coupled language) as  $L(G) = \{w \mid (S)^n \xrightarrow{*} (w_1, \dots, w_n), w_i \text{ is in } V_{T_i}^* \text{ and } w = w_1 \dots w_n\}$ . Similarly, we will define the n-translation generated by  $G$  as  $T_m(G) = \{(w, x) \mid (S)^n \xrightarrow{*} (w_1, \dots, w_n), w_i \text{ is in } V_{T_i}^*, w = w_1 \dots w_m, \text{ and } x = w_{m+1} \dots w_n\}$ .

We observe that the concept of a derivation in n-coupled grammars differs from that in grammars in the Chomsky hierarchy in that here, in a single step, particular occurrences of a nonterminal (which correspond with each other) are simultaneously

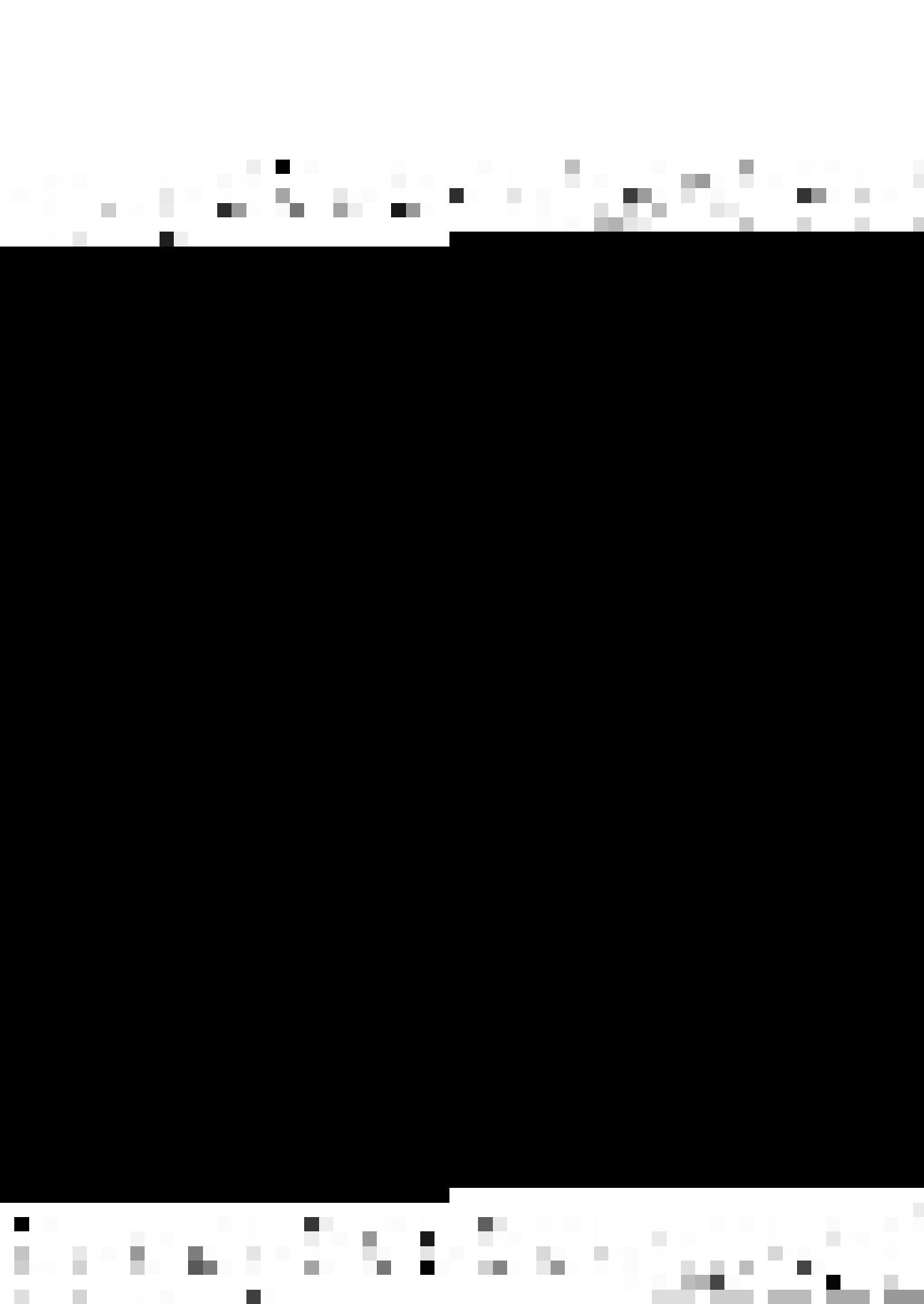
1. In section 1.2, we will be defining formally the languages and translations generated by coupled grammars and their various subclasses determined by certain structural constraints on the grammars generating them, in order to pave the way for their study in later chapters.



expanded. Effectively, it means letting the derivations in these grammars carry some contextual information, which is not the case in context-free grammars<sup>1</sup>. We get a feeling that these grammars should in general be more powerful than context-free grammars, which is in fact the case, as we shall see later. For discussion of the relationship of coupled languages (languages generated by coupled grammars) with some known classes of languages including the classes of context-free and context-sensitive languages, see Chapter 5.

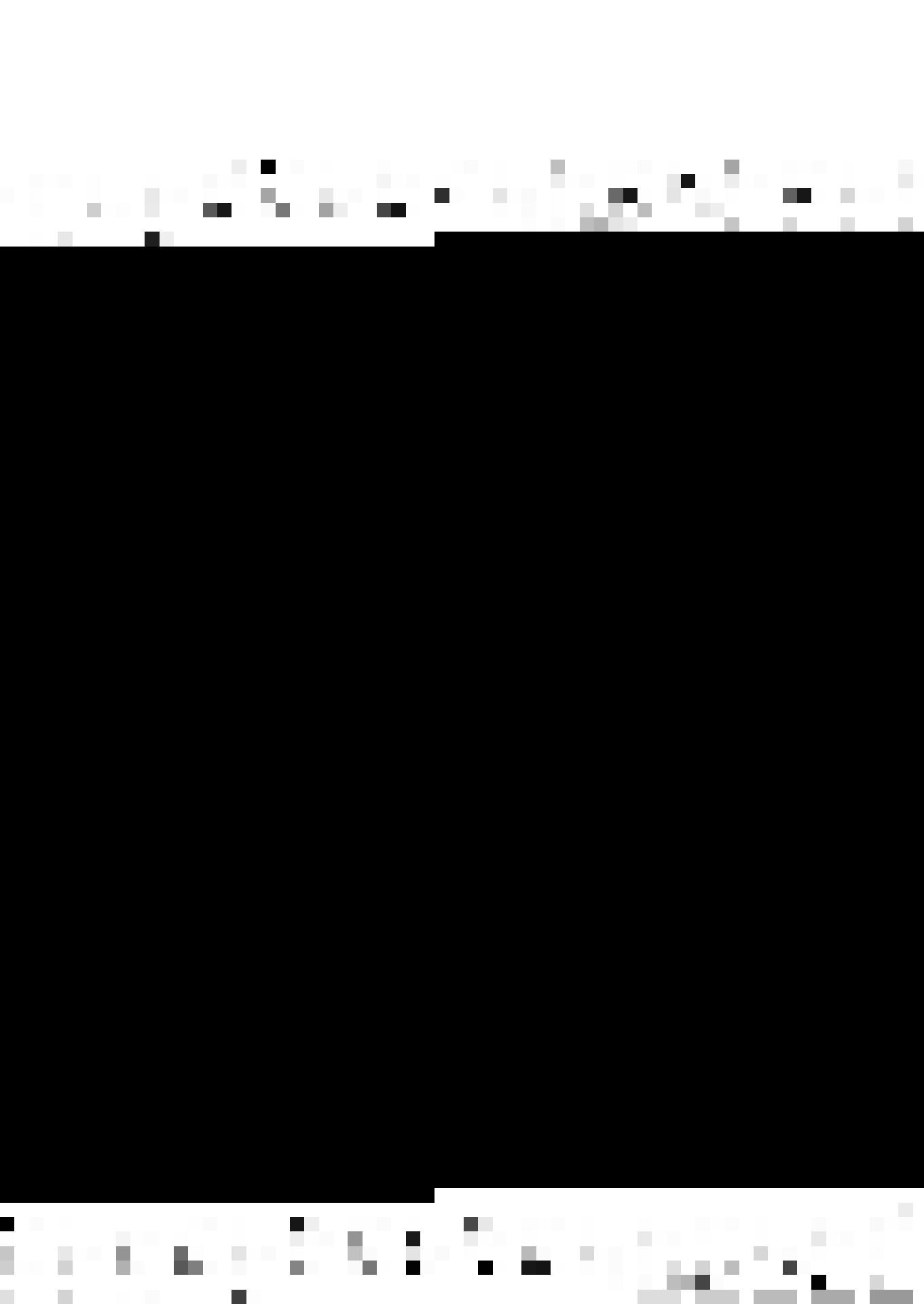
In Chapter 2, we are going to investigate the structure of coupled languages and coupled translations (translations generated by coupled grammars) — we are going to investigate whether the degree of coupled grammars and certain restrictions on the types of productions in these grammars induce structure on coupled languages and coupled translations. In order to pave the ground for it, we will define some restricted types of coupled grammars and certain parameters which indicate the complexity of coupled grammars. Before we do so, let us put across some ideas:

1. Observe that the contextual information carried in a derivation in an n-coupled grammar is not as much as is the case for context-sensitive grammars. The definition of 'absolutely parallel grammars' (30), 'scattered grammars' (17) and 'simple matrix grammars' (19) and the languages generated by them, are similar attempts at defining classes of languages which fall between the classes of context-free and context-sensitive languages.



We know that the right hand side of a production in an  $n$ -coupled grammar is a form in the grammar. A form in the grammar is an  $n$ -tuple in which each component may contain a permutation also. These permutations just serve the purpose of specifying which nonterminals correspond with which nonterminals in the form. Of course, here we can't draw any parallel with context-free grammars. If in the right hand side of each production in an  $n$ -coupled grammar, all the permutations involved are the same permutations, then we have a coupled grammar which is a 'simple coupled grammar'. The significance of this statement is that if we look at the  $n$ -tuple comprising the right hand side of a nonterminating production, the first nonterminals (from the left) in all the  $n$ 'strings' in the  $n$  components of the  $n$ -tuple, correspond with each other. Similarly, the second nonterminals in all the  $n$  'strings' correspond with each other and so on. Certainly such coupled grammars are 'simple' coupled grammars.

Again, given a coupled grammar, can we say something more than saying whether or not it is a simple coupled grammar i.e., can we define some parameter which indicates the 'simplicity' of the coupled grammar? We do make an attempt in this direction.



In context-free grammars, having more than two nonterminals in the right hand sides of the productions does not give them any more power than having atmost two nonterminals(18). We will see that this is not the case for coupled grammars. Therefore, we should define a parameter which tells what is the maximum number of nonterminals in the forms (in any component of any form) which are the right hand sides of the productions in the grammar.

Based on the above ideas, let us give the following definitions:

The  $\mu$ -simplicity<sup>1</sup> of every 1CG is  $\emptyset$  by convention. The  $\mu$ -simplicity of an nCG,  $n \geq 2$ ,  $G = (V_N, V_T, P, (S)^n)$ , is defined as the sequence  $\xi_1, \dots, \xi_{n-1}$ , where  $\xi_i$ ,  $1 \leq i \leq n-1$ , is the largest subset of  $\{i+1, \dots, n\}$  such that if  $\xi_i = \{i_1, \dots, i_s\}$ , then for all nonterminating productions:

$A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$  in  $P$ , each of  $\pi_{i_1}, \dots, \pi_{i_s}$  is the same permutation as  $\pi_i$ . If there is no nonterminating production in the grammar, then its  $\mu$ -simplicity is:  $\{2, \dots, n\}, \{3, \dots, n\}, \dots, \{n\}$ , by convention. \*\*\*

- 
1.  $\mu$ -simplicity is actually one of the complexity measures of coupled grammars similarly as degree is or order (which we define shortly) is.



If we think of the first components of the forms which are the right hand sides of the nonterminating productions as comprising the first 'part' of the productions, of the second components as comprising the second 'part' of the productions, and so on, the idea of the above definition of  $\mathcal{U}$ -simplicity is to indicate for each  $i$ ,  $1 \leq i \leq n - 1$ , which of the parts  $i + 1, \dots, n$  are 'simple' with respect to the part  $i$  and which are not.

The order  $k$  of an nCG  $G = (V_N, V_T, P, (S)^n)$  is the least integer such that for all nonterminating productions:  $A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$ , in  $P$ , the number of nonterminals in  $\alpha_1$  (and hence in each  $\alpha_i$ ,  $1 \leq i \leq n$ ), is no more than  $k$ ; if there is no nonterminating production in the grammar, then its order is said to be zero.

An nCG of  $\mathcal{U}$ -simplicity  $\sigma$  and order  $k$  is called an  $(n, \sigma, k)$ -coupled grammar  $((n, \sigma, k)CG)$ .  $G$  is an  $(n, \sigma)$ -coupled grammar  $((n, \sigma)CG)$  iff it is an  $(n, \sigma, k)CG$  for some value of  $k$ . Similarly,  $G$  is an  $(n, k)$ -coupled grammar  $((n, k)CG)$  iff it is an  $(n, \sigma, k)CG$  for some  $\sigma$ . An  $(n, k)$ -simple coupled grammar  $((n, k)SCG)$  is an  $(n, \sigma, k)CG$ , where for  $\sigma = \xi_1, \dots, \xi_n$   $\xi_1 = \{2, \dots, n\}^1$ . An  $n$ -simple coupled grammar ( $nSCG$ ) is an  $(n, k)SCG$  for some value of  $k$ . A simple coupled grammar ( $SCG$ ) is an  $nSCG$  for some value of  $n$ .

\*\*\*

---

1. It follows that  $\xi_i, 1 \leq i \leq n - 1 = \{i + 1, \dots, n\}$ .



We can put restrictions on the type of productions on the grammar in the same manner as restrictions are put on the types of productions in context-free grammars in order to define right linear grammars. Let us call the n-coupled grammars restricted in this manner as n-right linear coupled grammars. We shall show in Chapter 5 that n-right linear coupled grammars are equivalent in their generative power to equal matrix grammars of order n (34). Let us define formally this subclass of n-coupled grammars:

If  $G = (V_H, V_T, P, (S)^n)$  is an nCG such that the productions in  $P$  are only of the type:

$A \rightarrow (x_1B, \dots, x_nB)$ , or  $A \rightarrow (w_1, \dots, w_n)$ ,  $x_i, w_i$  in  $V_T^*$ ,  $A, B$ , in  $V_H$ , then  $G$  is called an n-right linear coupled grammar (nRLCG).

### Example 1.1.3:

The following is an example of a (2, 1)SCG i.e., a 2-simple coupled grammar of order 1:

$G = (\{S\}, \{a, b, c\}, P, (S)^2)$ , where  $P$  comprises of the productions:

1.  $S \rightarrow (aSb, cS)$ , and
2.  $S \rightarrow (ab, c)$ .



Example 1.1.4.

Again, the following is an example of a  $(3, \{3\}, \emptyset, 2)$ CG i.e., a coupled grammar of tree 3, of order 2, and of  $\mu$ -simplicity  $\{3\}, \emptyset$ .

$$G = (\{S, A, B\}, \{a_1, \dots, a_5, b_1, \dots, b_6\}, P, (S)^3),$$

where P comprises of the productions:

1.  $S \rightarrow (AB, (BA, [2, 1]), AB),$
2.  $A \rightarrow (a_1 b_1, a_4 A b_4, a_5 A b_5),$
3.  $B \rightarrow (a_2 B b_2, a_3 B b_3, a_6 B b_6),$
4.  $a \rightarrow (a_1 b_1, a_4 b_4, a_5 b_5),$  and
5.  $b \rightarrow (a_2 b_2, a_3 b_3, a_6 b_6).$

\*\*\*

Now, we come to an interesting problem. Given  $n$  and  $k$ , it looks intuitively that there should be some restrictions which  $\sigma$  should satisfy in order that an  $(n, \sigma, k)$ CG be realizable. Can we give necessary and sufficient conditions which  $\sigma$  should satisfy in order that an  $(n, \sigma, k)$ CG be realizable? The following theorem provides the required necessary and sufficient conditions.

Theorem 1.1.1.

(a)<sup>1</sup> For  $k = 1, n \geq 2$ , an  $(n, \sigma, k)$ CG is realizable iff for  $\sigma = \xi_1, \dots, \xi_{n-1}$ ,  $\xi_i$  is equal to  $\{i+1, \dots, n\}.$



(b) For  $k \geq 2$ ,  $n \geq 2$ ,  $\sigma = \xi_1, \dots, \xi_{n-1}$ , where  $\xi_i$  is some subset of  $\{i+1, \dots, n\}$ , an  $(n, \sigma, k)$ -CG is realizable iff for some  $s$ ,  $1 \leq s \leq n-1$ , and  $m > 1$ , none of the following statements is true:

1.  $l$  is in  $\xi_s$ ,  $m$  in  $\{s+1, \dots, n\}$  is not in  $\xi_s$  and  $m$  is in  $\xi_l$ .
2.  $l$  is in  $\xi_s$ ,  $m$  is in  $\xi_s$ , and  $m$  is not in  $\xi_l$ .
3.  $l$  in  $\{s+1, \dots, n\}$  is not in  $\xi_s$ ,  $m$  is in  $\xi_s$  and  $m$  is in  $\xi_l$ .

Proof:

(a) This follows from the fact that every  $(n, 1)$ -coupled grammar is by definition an  $(n, 1)$ -simple coupled grammar.

(b) only if:

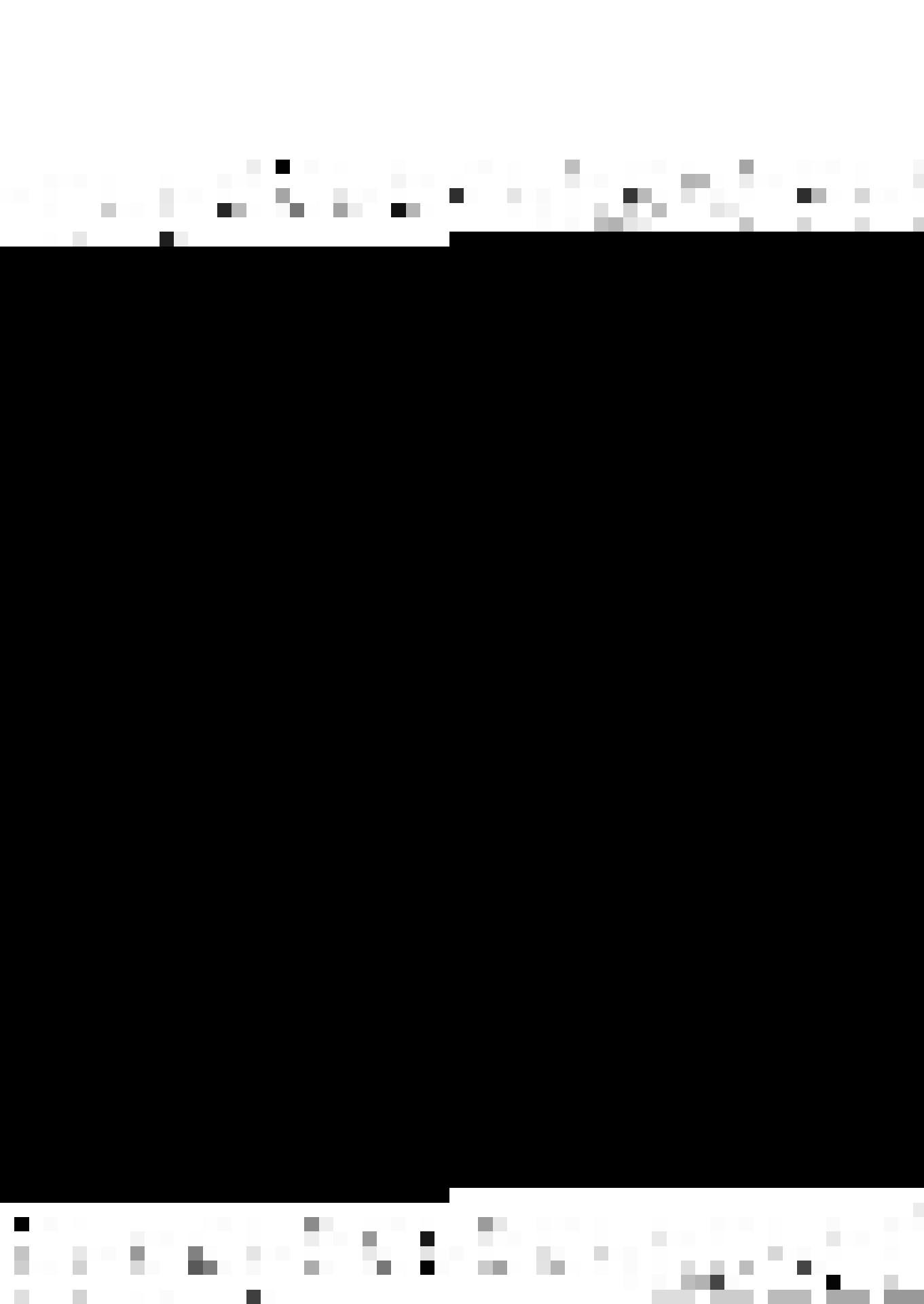
Let us suppose to the contrary, for the time being.

The idea is to get a contradiction using the definition of  $\mathcal{U}$ -simplicity.

Suppose there exists an  $(n, \sigma, k)$ -CG  $G = (V_N, V_T, P, (S)^n)$  for which statement 1 is true.

Now,  $l$  is in  $\xi_s$  implies that for all nonterminating productions:

$A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$ , in  $P$ ,  $\pi_1$  is the same permutation as  $\pi_s$ .  $m$  in  $\{s+1, \dots, n\}$  is not in  $\xi_s$  implies that for atleast one production:



$B \rightarrow ((\beta_1, \pi'_1), \dots, (\beta_n, \pi'_n))$ , in  $P$ ,  $\pi'_m$  is different from  $\pi'_s$ . Again,  $m$  is in  $\xi_1$  implies that for all nonterminating productions:

$A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$ , in  $P$ ,  $\pi_m$  is the same permutation as  $\pi_1$ . This clearly leads to a contradiction.

The proof, when statement 2 or 3 is true is similar to the above and is omitted.

if:

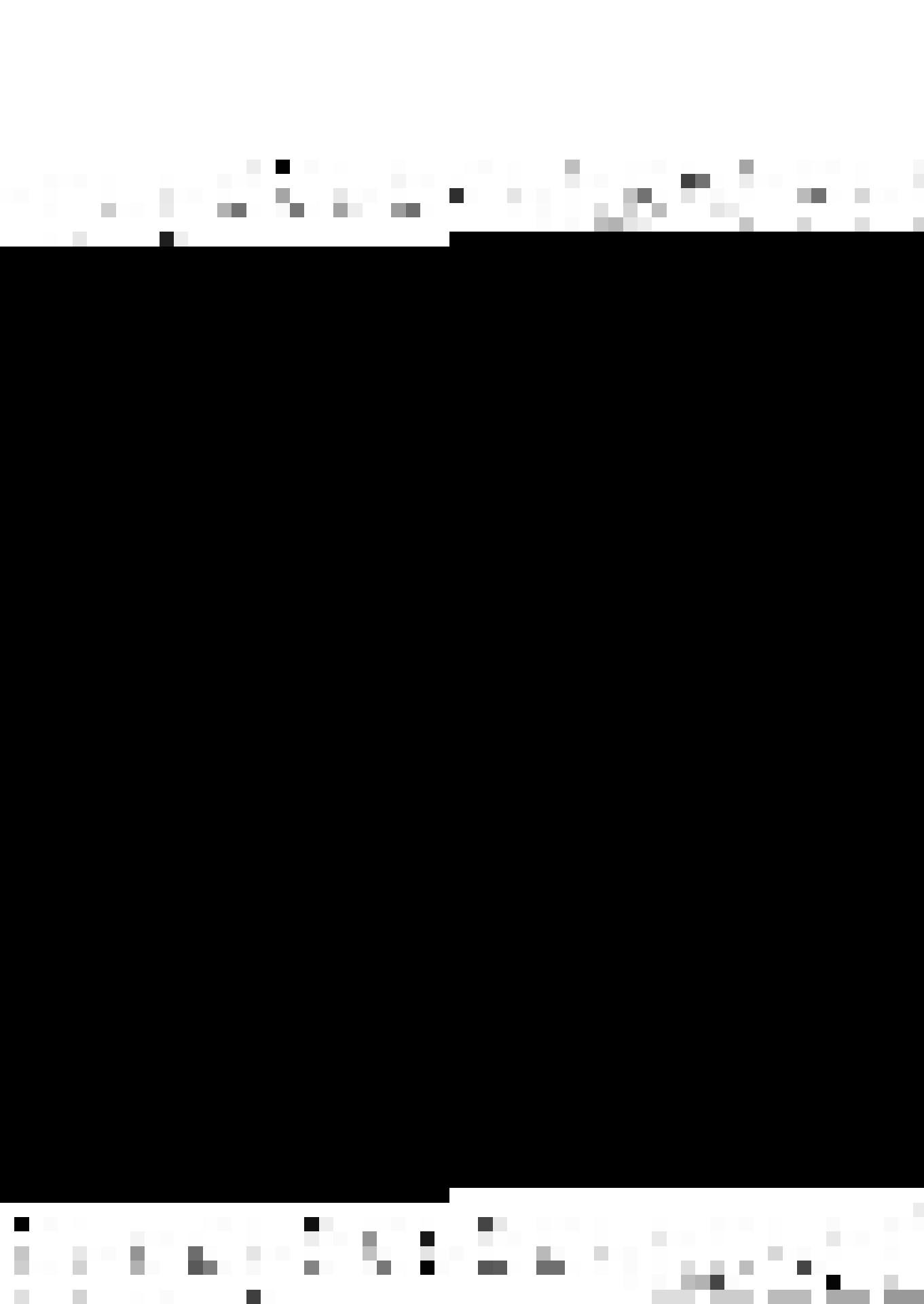
Notice that the three statements 1, 2, and 3 exhaust all the cases which can lead to a contradiction, among the eight possible cases:  $l$  may or may not be in  $\xi_s$ ,  $m$  may or may not be in  $\xi_1$ , and  $m$  may or may not be in  $\xi_s$ . This is the basic idea behind the following formal proof.

Let  $\sigma$  be such that none of the statements 1, 2, and 3 is true. In such a case, using the following algorithm, we shall construct an  $(n, \sigma, k)$ CG  $G$ , which we shall call, for the purpose of further reference, as the sample  $(n, \sigma, k)$ CG. This algorithm holds for  $n = 1$  as well; of course,  $k$  is greater than or equal to 2:

Algorithm:

$G = (\{A_1, \dots, A_k, B_1, B_2, S\}, \emptyset, P, (S)^n)$ , where  $P$  is constructed as follows:

(i) Let  $X = \{i_1, \dots, i_t\}$  such that  $X = \{2, \dots, n\} - \xi_1$  and the elements of  $X$  are in natural order.



$\mathbb{P} = \emptyset$ , to start with.

(ii) Add the production:

$$S \rightarrow (A_1 \dots A_n)^n, \text{ to } \mathbb{P}.$$

(This is to ensure that the order of  $G$  is  $k$ ).

(iii) Is  $X$  empty? If it is, then the algorithm terminates.

(iv) Let  $i_f$  be the first element in  $X$ . If  $i_f = n$  then let  $\xi_{i_f} = \emptyset$ . Let  $Y = (X \setminus \xi_{i_f}) \cup \{i_f\}$ .

(As a result,  $Y$  contains  $i_f$ , which is the smallest element in  $X$ , and all  $i$  such that  $i$  is in  $X$ , as well as in  $\xi_{i_f}$ . Actually we shall see that it cannot be that  $i$  is in  $\xi_{i_f}$  but is not in  $X$ .)

(v) Let  $X \leftarrow X - Y^1$ . Arrange the elements of  $X$  in natural order. Add to  $\mathbb{P}$  the production:

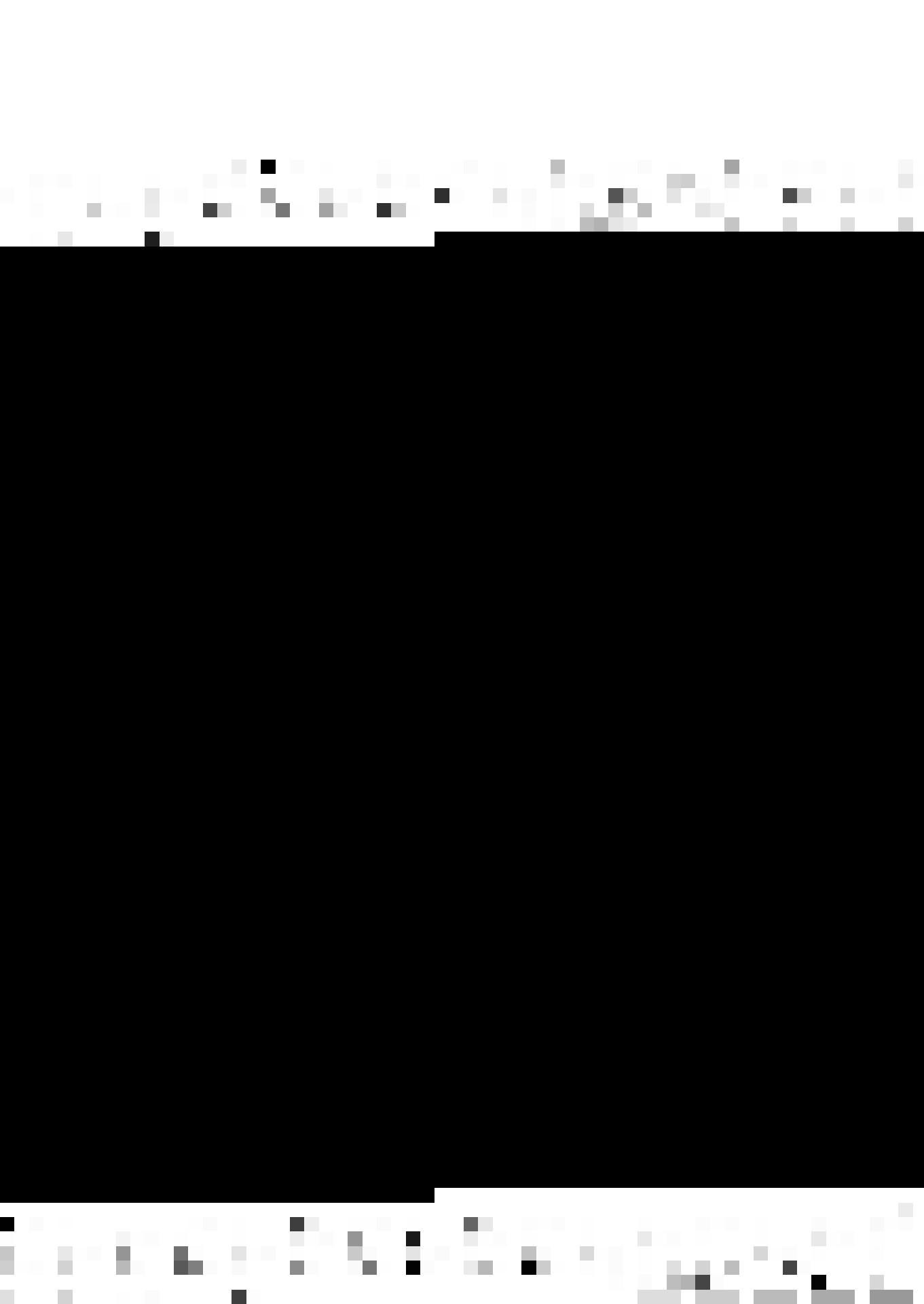
$S \rightarrow (B_1 B_2, (\alpha_2, \kappa_2), \dots, (\alpha_n, \kappa_n))$ , where if  $i$  is in  $Y$ , then  $\alpha_i = BA$ , and  $\kappa_i = [2,1]$ ; otherwise  $\alpha_i = AB$  and  $\kappa_i$  is an identity permutation.

(This is to ensure that if  $\sigma' = \xi'_1, \dots, \xi'_{n-1}$ , is the  $\mathcal{U}$ -simplicity of  $G$  as it stands at this step, then  $\xi'_{i_f} = \xi_{i_f}$ ).

(vi) Go to (iii).

This is the end of the algorithm.

• '←' stands for 'is replaced by'.



Observe that  $G$  does not generate any language. However, the above algorithm does not, in any way, depend on this fact.

We shall now show that  $G$  is in fact an  $(n, \sigma, k)CG$  by proving a sequence of statements  $(S_1)$ ,  $(S_2)$ ,  $(S_3)$ ,  $(S_4)$  and  $(S_5)$ :

$(S_1)$ : If  $j$  is in  $\xi_i$ , then  $\xi_j = \xi_i - \{i + 1, \dots, j\}$ .

This is so because if some  $p$  is in  $\xi_j$ , then it is also in  $\xi_i$ , because otherwise statement 1 of the theorem would be true and on the other hand if  $p$  is in  $\xi_i - \{i + 1, \dots, j\}$ , then it is also in  $\xi_j$ , because otherwise statement 2 would be true.

$(S_2)$ : If at any stage of iteration some  $i$  is in  $\xi_{i_f}$ , then it cannot be in  $\xi_1$  and hence  $i$  has to be in  $X$  as defined in step (i) of the algorithm.

This is so because otherwise statement 3 would be true.

$(S_3)$ : It cannot be that at any stage of iteration, in step (iv) some  $i$  is in  $\xi_{i_f}$  but is not in  $X$ , as  $X$  stands in that step.

Suppose this is not the case, then in view of  $(S_2)$  and the nature of steps (iv) and (v), it follows that for some  $j$ ,  $i_f$  is not in  $\xi_j$  but  $i$  is in  $\xi_j$ . This is, in fact, statement 3 and hence the contradiction.



$(S_4)$ : Let  $i$  in  $\{2, \dots, n\}$  be not in  $\xi_1$  and let  $\xi_i = \{i_1, \dots, i_s\}$ . There is a production in  $P$ , say

$\cancel{\gamma}$ :  $S \rightarrow (\alpha_1, (\alpha_2, \bar{\kappa}_2), \dots, (\alpha_n, \bar{\kappa}_n))$ , in which  $\bar{\kappa}_i = \bar{\kappa}_{i_1} = \dots = \bar{\kappa}_{i_s} = [2, 1]$ ,  $\bar{\kappa}_j$  is identity for  $j$  in  $\{i+1, \dots, n\}$  but not in  $\xi_i$ ; and for any other production in  $P$ , say:

$S \rightarrow (\alpha'_1, (\alpha'_2, \bar{\kappa}'_2), \dots, (\alpha'_n, \bar{\kappa}'_n))$ ,  $\bar{\kappa}'_i, \bar{\kappa}'_{i_1}, \dots, \bar{\kappa}'_{i_s}$  are identity permutations.

Note that at a certain stage of iteration, in step (iv), either  $i = i_f$  or  $i$  is in  $\xi_{i_f}$ , for some  $i_f$ .

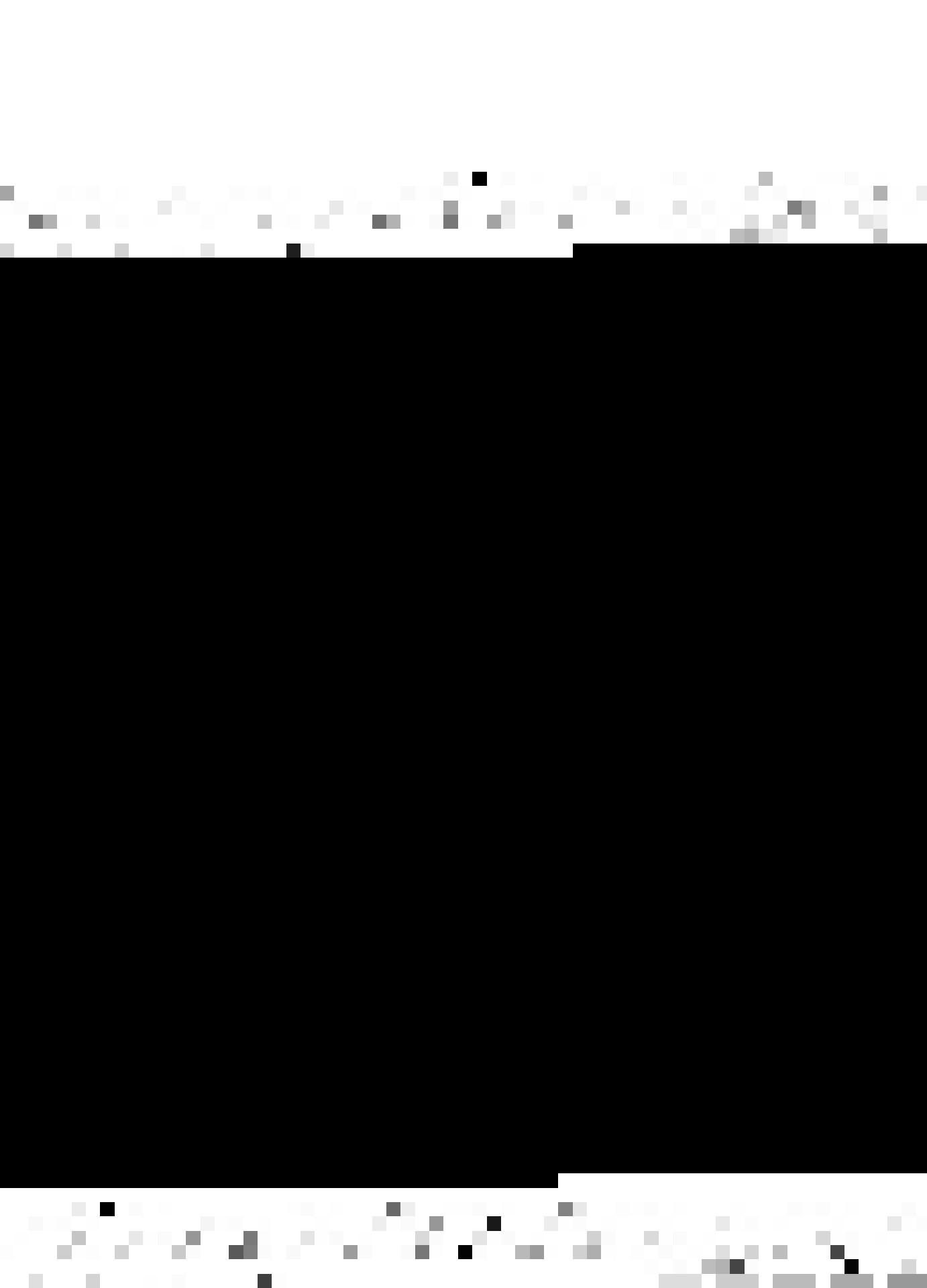
Again, in view of  $(S_3)$ , step (iv) can be reframed as:  
Let  $i_f$  be the first element in  $X$ . Let  $Y = \xi_{i_f} \cup \{i_f\}$ .

It now follows that the production introduced in the immediately next step is  $\cancel{\gamma}$  (in case  $i \neq i_f$ , this is so in view of  $(S_1)$ ).

The later part of  $(S_4)$  follows in view of the nature of steps (iv) and (v), i.e., the way  $Y$  is constructed, the subtraction of the set  $Y$  from  $X$ , and the nature of the productions introduced in step (v).

$(S_5)$ : If  $i$  is in  $\xi_1$ , then for all productions:

$S \rightarrow (\alpha_1, (\alpha_2, \bar{\kappa}_2), \dots, (\alpha_n, \bar{\kappa}_n))$ , in  $P$ ,  $\bar{\kappa}_i$  is an identity permutation.



Note that because  $i$  cannot be in  $X$  at any stage, it cannot be in  $Y$  as well, at any stage.  $(S_5)$  now follows in view of the nature of the productions introduced in steps (iii) and (v).

It now follows from  $(S_1)$ ,  $(S_4)$  and  $(S_5)$  that  $G$  is an  $(n, \sigma, k)$ CG.

Q.E.D.

### Example 1.1.5:

In this example, we shall illustrate the construction of the sample  $(n, \sigma, k)$ CG for  $n = 4$ ,  $k = 3$ , and  $\sigma = \{4\}, \{3\}$ , & using the algorithm given in the proof of the above theorem.

$G = (\{A_1, A_2, A_3, B_1, B_2, S\}, \emptyset, P, (S)^4)$ , where  $P$  is constructed as follows:

1.  $X = \{2, 3\}$ .  $P = \emptyset$ , to start with.

2. Add the production:

$S \rightarrow (A_1 A_2 A_3)^4$ , to  $P$ .

3.  $Y = \{2, 3\}$ .  $X \leftarrow X - Y$ . Add to  $P$  the production:

$S \rightarrow (B_1 B_2, (B_2 B_1, [2, 1]), (B_2 B_1, [2, 1]), B_1 B_2)$ .

4. The algorithm terminates because  $X = \emptyset$ .

Thus,  $P$  comprises of the productions:

$S \rightarrow (A_1 A_2 A_3)^4$ , and

$S \rightarrow (B_1 B_2, (B_2 B_1, [2, 1]), (B_2 B_1, [2, 1]), B_1 B_2)$ .

Clearly,  $G$  is  $(4, \{4\}, \{3\}, \emptyset, 3)$ CG.

\*\*\*



It certainly does not make any sense to talk of those  $(n, \sigma, k)CG$ 's (and the translations and languages generated by such grammars) which just cannot be realized. We therefore have the convention:

#### Convention 1.1.1:

From now onwards, an  $(n, \sigma, k)CG$  would always mean a realizable  $(n, \sigma, k)CG$ . We will be concerned, in this thesis, with realizable  $(n, \sigma, k)CG$ 's only.

#### 1.2 COUPLED TRANSLATIONS AND COUPLED LANGUAGES:

In this section, we set ourselves to formally defining the translations and languages generated by coupled grammars and some of their subclasses.

Let  $T$  be a translation<sup>1</sup> from a finite input alphabet  $V_i$  to a finite output alphabet  $V_o$ . Any element of  $T$  will be denoted by  $(w, x)$ , where  $w$  is in  $V_i^*$  and  $x$  is in  $V_o^*$ . The domain  $D$  of  $T$  is defined as  $\{w \mid \text{for some } x \text{ in } V_o^*, (w, x) \text{ is in } T\}$ . The range  $R$  of  $T$  is defined as  $\{x \mid \text{for some } w \text{ in } V_i^*, (w, x) \text{ is in } T\}$ .

\*\*\*

Because of our concern in the next chapter with the investigation of the structure of coupled languages and coupled translations induced by the  $\lambda$ -simplicity, order and the degree of the coupled grammars generating them, we will now define the

---

For the definitions of a 'translation', 'language' and related concepts, see (7).



$m$ -translations and the languages generated by  $(n, \sigma, k)$ -coupled grammars,  $1 \leq m \leq n - 1$  ( $m$  refers to the number of initial components of a terminal form derived from the start form, which are concatenated in order, to form  $w$ , such that  $(w, x)$  is an element of the translation generated).

Let  $G = (V_N, \{V_{T_1} \cup \dots \cup V_{T_n}\}, P, (S)^n)$  be an  $(n, \sigma, k)$ CG.

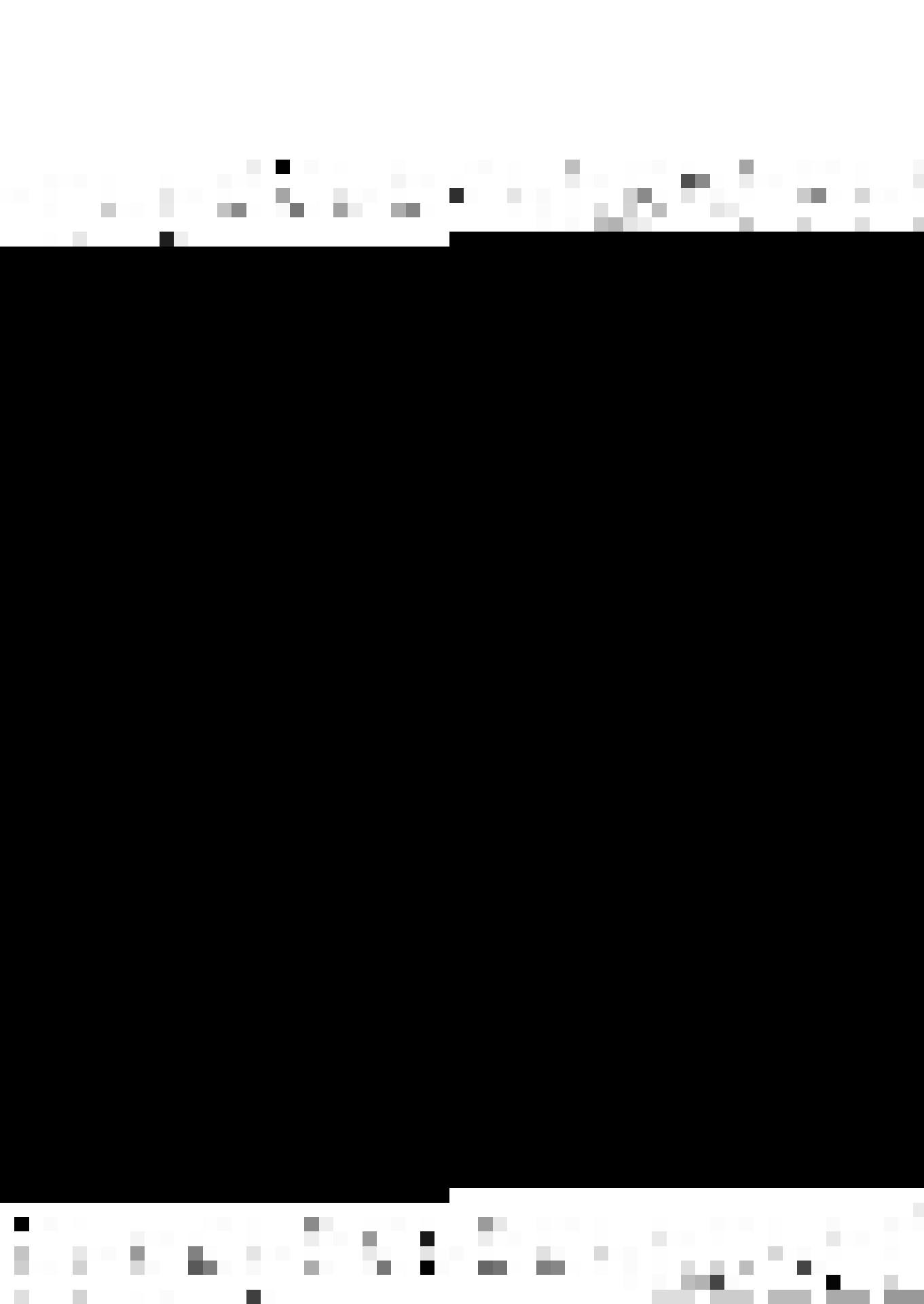
The  $m$ -translation generated by  $G$ ,  $1 \leq m \leq n - 1$ , from the input alphabet  $V_i = V_{T_1} \cup \dots \cup V_{T_m}$  to the output alphabet  $V_o = V_{T_{m+1}} \cup \dots \cup V_{T_n}$ , denoted by  $T_m(G)$  is:

$\{(w, x) | w \text{ is in } V_i^*, x \text{ is in } V_o^*, (S)^n \xrightarrow[G]{*} (w_1, \dots, w_m, x_1, \dots, x_{n-m}), w = w_1 \dots w_m \text{ and } x = x_1 \dots x_{n-m}\}$ .  
and is called an  $(m, n-m, \sigma, k)$ -coupled translation  $((m, n-m, \sigma, k)CT$ ).  
 $T$  is an  $(m, n-m, \sigma, k)CT$  iff  $T = T_m(G)$  for some  $(n, \sigma, k)CG$   $G$ .

The language generated by  $G$  over the alphabet  $I = V_T$ , denoted by  $L(G)$  is:

$\{w | w \text{ is in } V_T^*, (S)^n \xrightarrow[G]{*} (w_1, \dots, w_n), \text{ and } w = w_1 \dots w_n\}$ ,  
and is called an  $(n, \sigma, k)$ -coupled language  $((n, \sigma, k)CL)$ .

$T$  is an  $(m, n, \sigma, k)CT$ ,  $m \geq 1, n \geq 1$ , iff  $T = T_m(G)$  for some  $(m, n, \sigma, k)CG$   $G$ . Similarly,  $L$  is an  $(n, \sigma, k)CL$  iff  $L = L(G)$  for some  $(n, \sigma, k)CG$   $G$ .



We note that an  $(1,1,k)$ CT is the same as the 'syntax directed translation of order  $k$ ' defined in (3).

Let us now give notations and definitions for the various classes of translations and languages, which we will be interested in, in the rest of the thesis (all of which are, of course, subclasses of coupled translations and coupled languages, respectively).

$\mathcal{C}_{(m,n,\sigma,k)}^T$  denotes the class of  $(m,n,\sigma,k)$ -coupled translations. Omission of the parameter  $k$  in  $\mathcal{C}_{(m,n,\sigma,k)}^T$  denotes the union of this class over all possible values of  $k$ . Omission of  $\sigma$  denotes its union over all possible values of  $\sigma$  such that  $(m+n,\sigma,k)$ CG is realizable (See Theorem 1.1.1). In this manner we have the classes of translations  $\mathcal{C}_{(m,n,\sigma)}^T$  and  $\mathcal{C}_{(m,n,k)}^T$ .  $\mathcal{C}_{(m,n)}^T$  is the union of  $\mathcal{C}_{(m,n,k)}^T$  over all possible values of  $k$ .  $\mathcal{C}^T$  is the union of  $\mathcal{C}_{(m,n)}^T$  over all possible values of  $m$  and  $n$ .

$\mathcal{C}_{(n,\sigma,k)}^L$  denotes the class of  $(n,\sigma,k)$ -coupled languages. Similarly as above, omission of a parameter indicates the union over that parameter. Thus we have the classes of languages  $\mathcal{C}_{(n,\sigma)}^L$ ,  $\mathcal{C}_{(n,k)}^L$ ,  $\mathcal{C}_n^L$  and  $\mathcal{C}^L$ .

$T$  is an  $\alpha$ -coupled translation<sup>1</sup> ( $\alpha$ CT) iff it is in  $\mathcal{C}_\alpha^T$ .  
 (Notice that  $\alpha$  can have one of the values:  $(m,n,\sigma)$ ,  $(m,n,k)$ ,

- 
1. Notice that because of various contexts like:  $\mathcal{U}$ -simplicity is a sequence of sets, etc, there should not be any confusion about the terminology.



$(m, n)$ ). It is a coupled translation iff it is in  $\mathcal{C}^T$ .

$L$  is an  $\alpha$ -coupled language ( $\alpha CL$ ) iff it is in  $\mathcal{C}_\alpha^L$ .  
 (Notice that  $\alpha$  can have one of the values:  $(n, \sigma)$ ,  $(n, k), n$ ).  
 It is a coupled language iff it is in  $\mathcal{C}^L$ .

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, k) SCG$ . Then,  $T_m(G)$ ,  
 $1 \leq m \leq n-1$  is called an  $(m, n-m, k)$ -simple coupled translation  
 $((m, n-m, k) SCT)$ . Similarly,  $L(G)$  is called an  $(n, k)$ -simple  
coupled language  $((n, k) SCL)$ .

$\mathcal{S}_{(m, n, k)}^T$  denotes the class of  $(m, n, k)$ -simple coupled  
 translations.  $\mathcal{S}_{(n, k)}^L$  denotes the class of  $(n, k)$ -simple coupled  
 languages.

In  $\mathcal{S}_{(m, n, k)}^T$ , the parameter  $k$  or all the parameters  
 $m, n$ , and  $k$  may be omitted which would mean, similarly as  
 above, union over all possible values of the omitted para-  
 meter(s). Similarly, in  $\mathcal{S}_{(n, k)}^L$ ,  $k$  or both  $n$  and  $k$  may be  
 omitted which would mean union over all possible values of  
 the omitted parameter(s). Thus, we have  $\mathcal{S}_{(m, n)}^T$ ,  $\mathcal{S}^T$  as classes  
 of translations and  $\mathcal{S}_n^L$ ,  $\mathcal{S}^L$  as classes of languages.

$T$  is an  $(m, n)$ -simple coupled translation  $((m, n) SCT)$   
 iff it is in  $\mathcal{S}_{(m, n)}^T$  and is a simple coupled translation (SCT)  
 iff it is in  $\mathcal{S}^T$ .

$L$  is an  $n$ -simple coupled language ( $n SCL$ ) iff it is in  
 $\mathcal{S}_n^L$  and is a simple coupled language (SCL) iff it is in  $\mathcal{S}^L$ .



Let  $G = (V_N, \{v_{T_1} \cup \dots \cup v_{T_n}\}, P, (S)^n)$  be an nRLCG. Then,  $T_m(G)$  is called an  $(m, n-m)$ -right linear coupled translation ( $(m, n-m)$ RLCT),  $1 \leq m \leq n-1$ .

Similarly,  $L(G)$  is called an n-right linear coupled language (nRLCL).

$T$  is an  $(m, n)$ RLCT,  $m \geq 1, n \geq 1$ , iff  $T = T_m(G)$  for some  $(m+n)$ RLCG  $G$ . Similarly  $L$  is an nRLCL iff  $L = L(G)$  for some nRLCG  $G$ .

$\mathcal{R}_{(m, n)}^T$  denotes the class of  $(m, n)$ -right linear coupled translations.  $\mathcal{R}_n^L$  denotes the class of n-right linear coupled languages and  $\mathcal{R}^L$  denotes class of right linear coupled languages.

#### Example 1.2.1:

Let  $G$  be the  $(2, 1)$ SCG of Example 1.1.3. It is easy to see that  $T_1(G) = \{(a^i b^i, c^i) | i \geq 1\}$ . We observe that  $T_1(G)$  is an  $(1, 1, 1)$ SCT. Also, the  $(2, 1)$ SCL generated by  $G$  is  $L(G) = \{a^i b^i c^i | i \geq 1\}$ .

\*\*\*

It is well known (14) that the language  $\{a^i b^i c^i | i \geq 1\}$  is not a context-free language. Thus we have:

#### Lemma 1.2.1:

Not all languages in  $\mathcal{S}_{(2, 1)}^L$  are context-free.



Example 1.2.2:

Let  $G$  be the  $(3, \{3\}, \emptyset, 2)$ CG of Example 1.1.4. It is easy to see that

$$T_1(G) = \left\{ (a_1^{n_1} b_1^{n_1} a_2^{n_2} b_2^{n_2}, a_3^{n_2} b_3^{n_2} a_4^{n_1} b_4^{n_1} a_5^{n_1} b_5^{n_1} a_6^{n_2} b_6^{n_2}) | n_1, n_2 \geq 1 \right\},$$

$$T_2(G) = \left\{ (a_1^{n_1} b_1^{n_1} a_2^{n_2} b_2^{n_2} a_3^{n_2} b_3^{n_2} a_4^{n_1} b_4^{n_1}, a_5^{n_1} b_5^{n_1} a_6^{n_2} b_6^{n_2}) | n_1, n_2 \geq 1 \right\},$$

and

$$L(G) = \left\{ a_1^{n_1} b_1^{n_1} a_2^{n_2} b_2^{n_2} a_3^{n_2} b_3^{n_2} a_4^{n_1} b_4^{n_1} a_5^{n_1} b_5^{n_1} a_6^{n_2} b_6^{n_2} | n_1, n_2 \geq 1 \right\}.$$

We notice that  $T_1(G)$  is an  $(1, 2, \{3\}, \emptyset, 2)$ CT,  $T_2(G)$  is an  $(2, 1, \{3\}, \emptyset, 2)$ CT, and  $L(G)$  is an  $(3, \{3\}, \emptyset, 2)$ CL.

\*\*\*

We will now see that the domain as well as the range of coupled translations are coupled languages. In this connection, we will define<sup>1</sup> the ' $m$ -input grammar' and the ' $n$ -output grammar' of an  $(m+n, \sigma, k)$ CG. The idea behind these definitions is that the domain and the range of the  $m$ -translation generated by an  $(m+n, \sigma, k)$ CG  $G$  are generated by the  $m$ -input grammar and the  $n$ -output grammar of  $G$ , respectively.

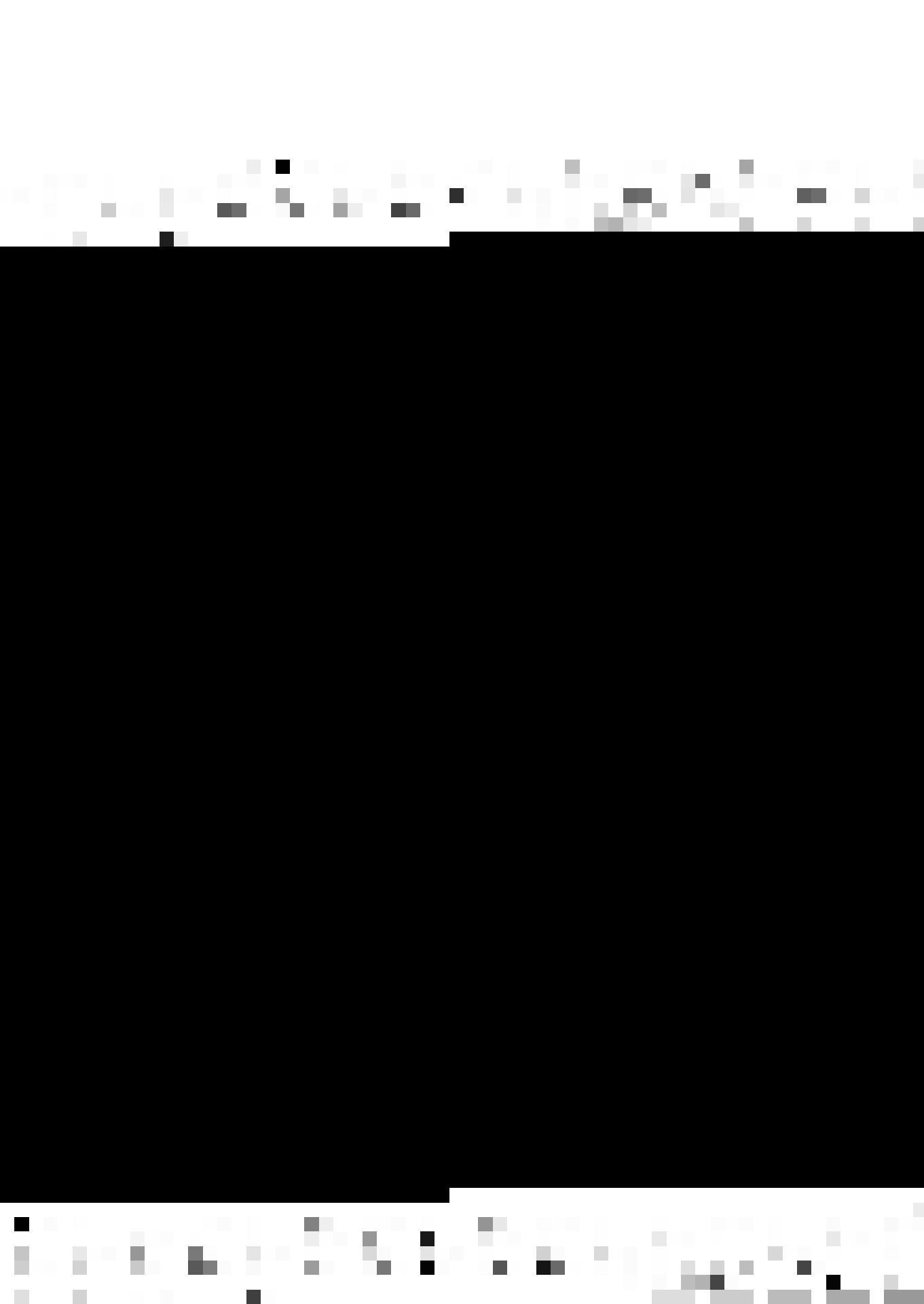
Let  $G = (V_N, V_T, \cup_{i=1}^m V_{T_i}, P, (S)^{m+n})$ , be an  $(m+n, \sigma, k)$ CG,  $m \geq 1, n \geq 1$ , where  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ .

The  $m$ -input grammar of  $G$  is defined as:

$$G_i = (V_N, V_{T_i}, \cup_{j=1}^m V_{T_j}, P_i, (S)^m), \text{ where}$$

$$P_i = \{ A \rightarrow ((\alpha_1, \star_1), \dots, (\alpha_m, \star_m)) \mid \text{a nonterminating} \}$$

1. We use these concepts later also.



production:  $A \rightarrow ((\alpha_1, \bar{\alpha}_1), \dots, (\alpha_{m+n}, \bar{\alpha}_{m+n}))$ , is in  $P\} \cup \{A \rightarrow (w_1, \dots, w_m) | \text{ there is a terminating production: } A \rightarrow (w_1, \dots, w_{m+n}), \text{ in } P\}.$

The n-output grammar of  $G$  is defined as:

$$G_o = (V_N, V_{T_{m+1}} \cup \dots \cup V_{T_{m+n}}, P_o, (S)^n),$$

where,

$P_o = \{A \rightarrow ((\alpha_{m+1}, \bar{\alpha}_{m+1}), \dots, (\alpha_{m+n}, \bar{\alpha}_{m+n})) | \text{ there is a nonterminating production: } A \rightarrow ((\alpha_1, \bar{\alpha}_1), \dots, (\alpha_{m+n}, \bar{\alpha}_{m+n})), \text{ in } P\} \cup \{A \rightarrow (w_{m+1}, \dots, w_{m+n}) | \text{ there is a terminating production: } A \rightarrow (w_1, \dots, w_{m+n}), \text{ in } P\}.$

\*\*\*

We observe that the order of  $G_i$  is  $k$  and the  $\mathcal{U}$ -simplicity of  $G_i$  is  $\sigma_i = \xi_1 \cap \{2, \dots, m\}, \dots, \xi_{m-1} \cap \{2, \dots, m\}$ . Also, we observe that the order of  $G_o$  is  $k$  and if its  $\mathcal{U}$ -simplicity is  $\sigma_o = \xi'_1, \dots, \xi'_{n-1}$ , then

$$\xi'_i = \{j | m+j \text{ is in } \xi_{n+i}\}, 1 \leq i \leq n-1.$$

It is easy to prove the following lemma; we will omit the proof:

### Lemma 1.2.2:

Let  $G = (V_N, V_{T_1} \cup \dots \cup V_{T_{m+n}}, P, (S)^{m+n})$ , be an  $(m+n, \sigma, k)$ CG such that  $T_m(G)$  is the given  $(m, n, \sigma, k)$ CT  $T$ . Let  $G_i$  and  $G_o$  be the  $m$ -input and the  $n$ -output grammar of  $G$ , respectively. Then, the domain of  $T$  is  $L(G_i)$  and the range of  $T$  is  $L(G_o)$ .

\*\*\*



We thus have:

Theorem 1.2.1:

Let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ .

(a) The domain of every  $(m, n, \sigma, k)CT$  is an  $(m, \sigma_i, k)CL$ ,

where  $\sigma_i = \xi_1 \cap \{2, \dots, m\}, \dots, \xi_{m-1} \cap \{2, \dots, m\}$ .

(b) The range of every  $(m, n, \sigma, k)CT$  is an  $(n, \sigma_o, k)CL$ ,

where if  $\sigma_o = \xi'_1, \dots, \xi'_{n-1}$ , then  $\xi'_i = \{j \mid m+j \text{ is in } \xi_{m+i}\}, 1 \leq i \leq n-1$ .

\*\*\*

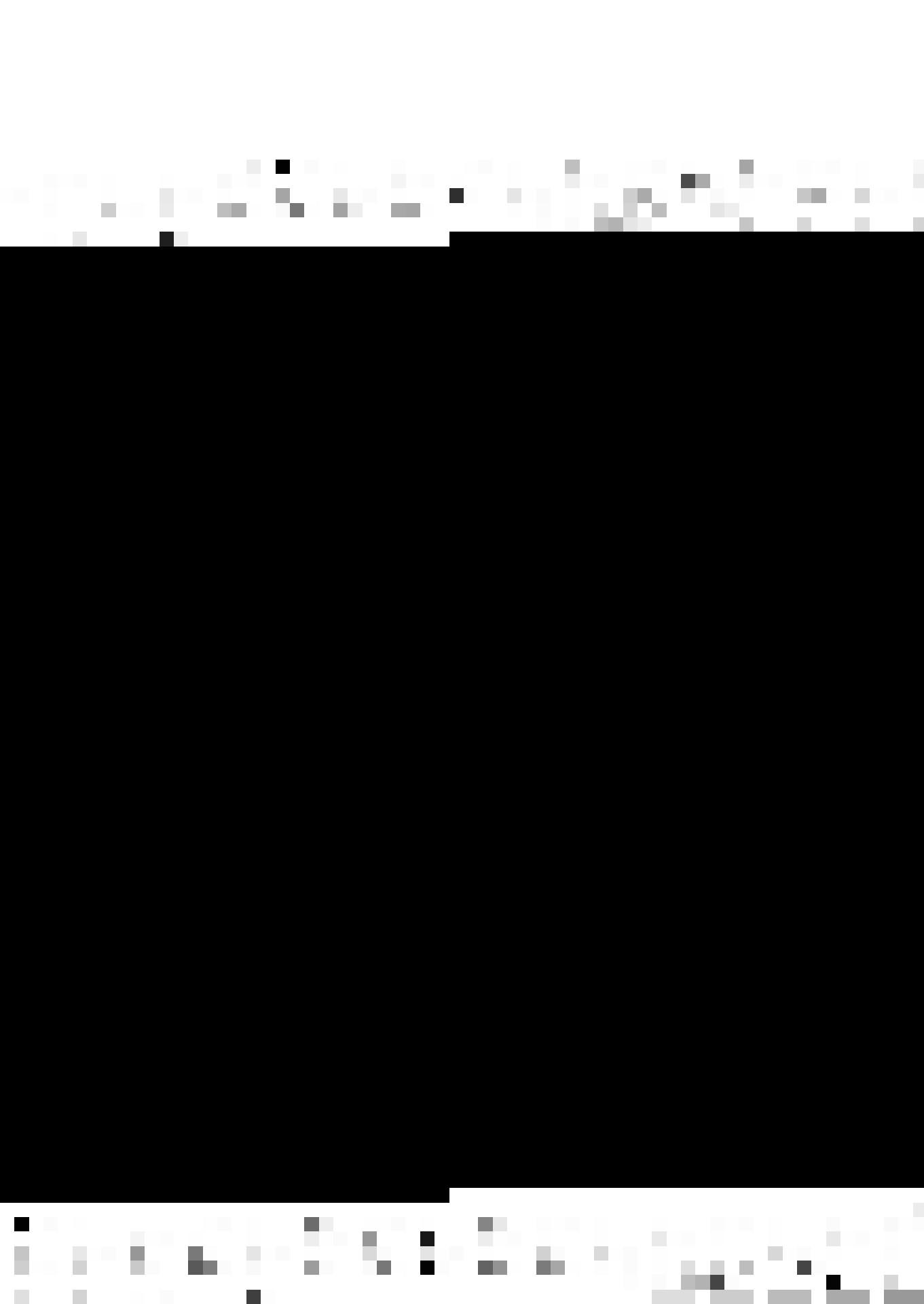
We now come to a lemma which will justify a convention that we will usually follow in the rest of the thesis. This is about the fact that having the permutation in the first component of a form in an  $n$ -coupled grammar as a non-identity permutation does not convey any more information about the form than having it as an identity permutation.

Lemma 1.2.3:

Given an  $(n, \sigma, k)CG G = (V_N, V_T, P, (S)^n)$  such that for some nonterminating production:

$A \rightarrow ((\alpha_1, \kappa_1), \dots, (\alpha_n, \kappa_n))$ , in  $P$ ,  $\kappa_1$  is not an identity permutation, we can construct an  $(n, \sigma, k)CG G' = (V_N, V_T, P', (S)^n)$  such that all nonterminating productions in  $P'$  are of the type:

$A \rightarrow (\alpha_1, (\alpha_2, \kappa_2), \dots, (\alpha_n, \kappa_n))$ , and  $T_i(G) = T_i(G')$  for all  $i$ ,  $1 \leq i \leq n-1$  (and hence  $L(G) = L(G')$ ).



Proof:

The construction of  $G'$  from  $G$  is as follows: The terminating productions in  $P'$  are the same as those in  $P$ . The nonterminating productions in  $P'$  are constructed as follows:

- (i) Add to  $P'$  all nonterminating productions of  $P$  of the type:  $A \rightarrow (\alpha_1, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n))$ ,
- (ii) For each nonterminating production:  $A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$  in  $P$  such that  $\pi_1$  is not an identity permutation, add to  $P'$  the production:

$A \rightarrow (\alpha_1, (\alpha_2, \pi'_2), \dots, (\alpha_n, \pi'_n))$ , where if  $\pi_1$  is a permutation on  $m$  objects then for all  $i$ ,  $2 \leq i \leq n$ ,

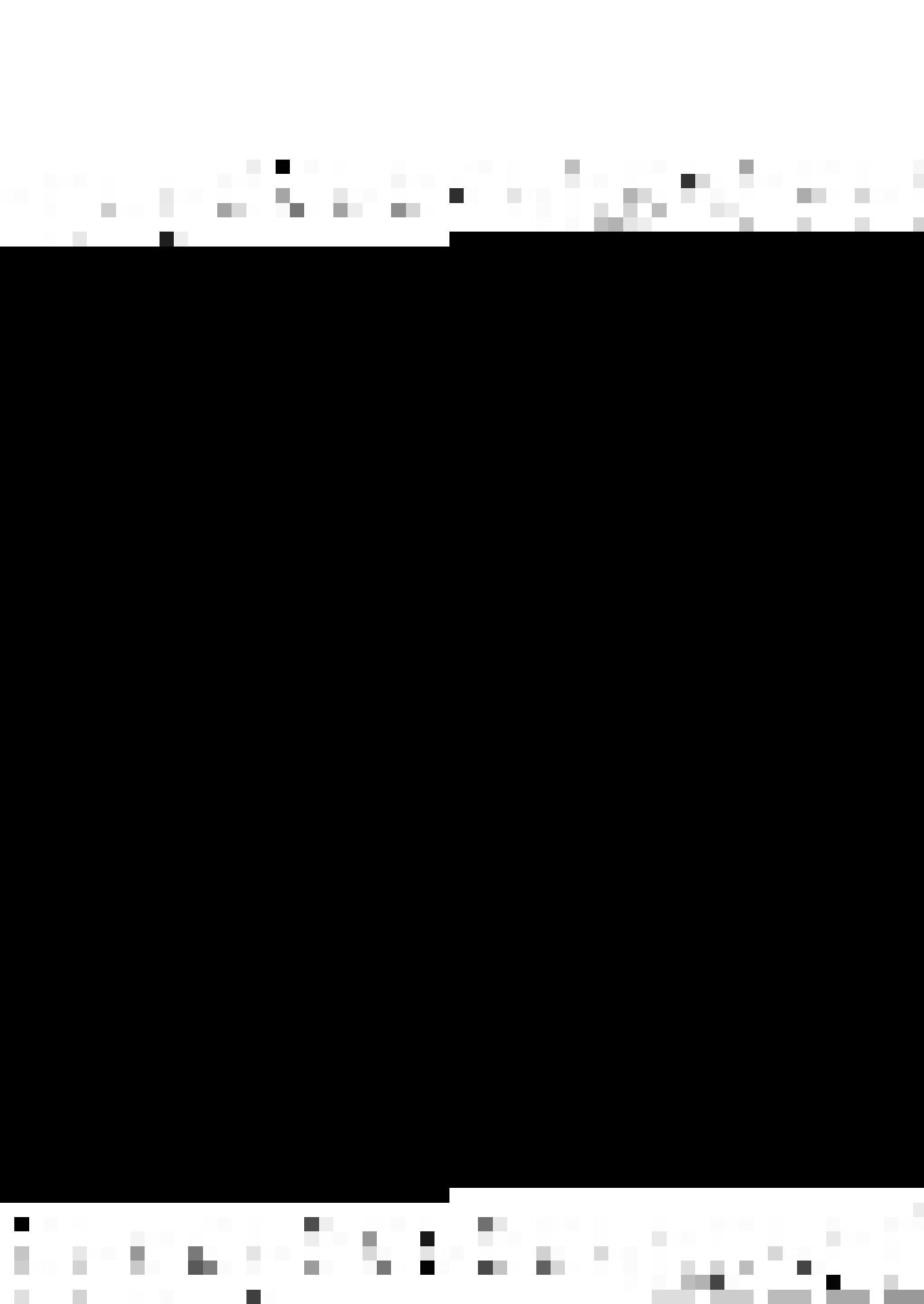
$$\pi'_i = [\pi_i(\hat{\pi}_1(1)), \dots, \pi_i(\hat{\pi}_1(m))].$$

Observe that the definition of  $\pi'_i$  is such that the correspondence between the nonterminals in  $\alpha_1, \dots, \alpha_n$  in the form:  $(\alpha_1, (\alpha_2, \pi'_2), \dots, (\alpha_n, \pi'_n))$  (which is a form in  $G'$ ) is the same as that between the nonterminals in  $\alpha_1, \dots, \alpha_n$ , in the form:

$$((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n)) \text{ (which is a form in } G).$$

It is obvious that  $G'$  is an  $(n, \sigma, k)$ CG. The result follows now.

Q.E.D.



Example 1.2.3:

We shall illustrate how, in the above lemma,  $G'$  is constructed from  $G$ .

Let  $G$  be the grammar given in Example 1.1.1 viz.,  
 $G = (V_N, V_T, P, (S)^3)$ , where  $V_N = \{S, A, B\}$ ,  $V_T = \{a, b\}$ , and  
 $P$  comprises of the productions:

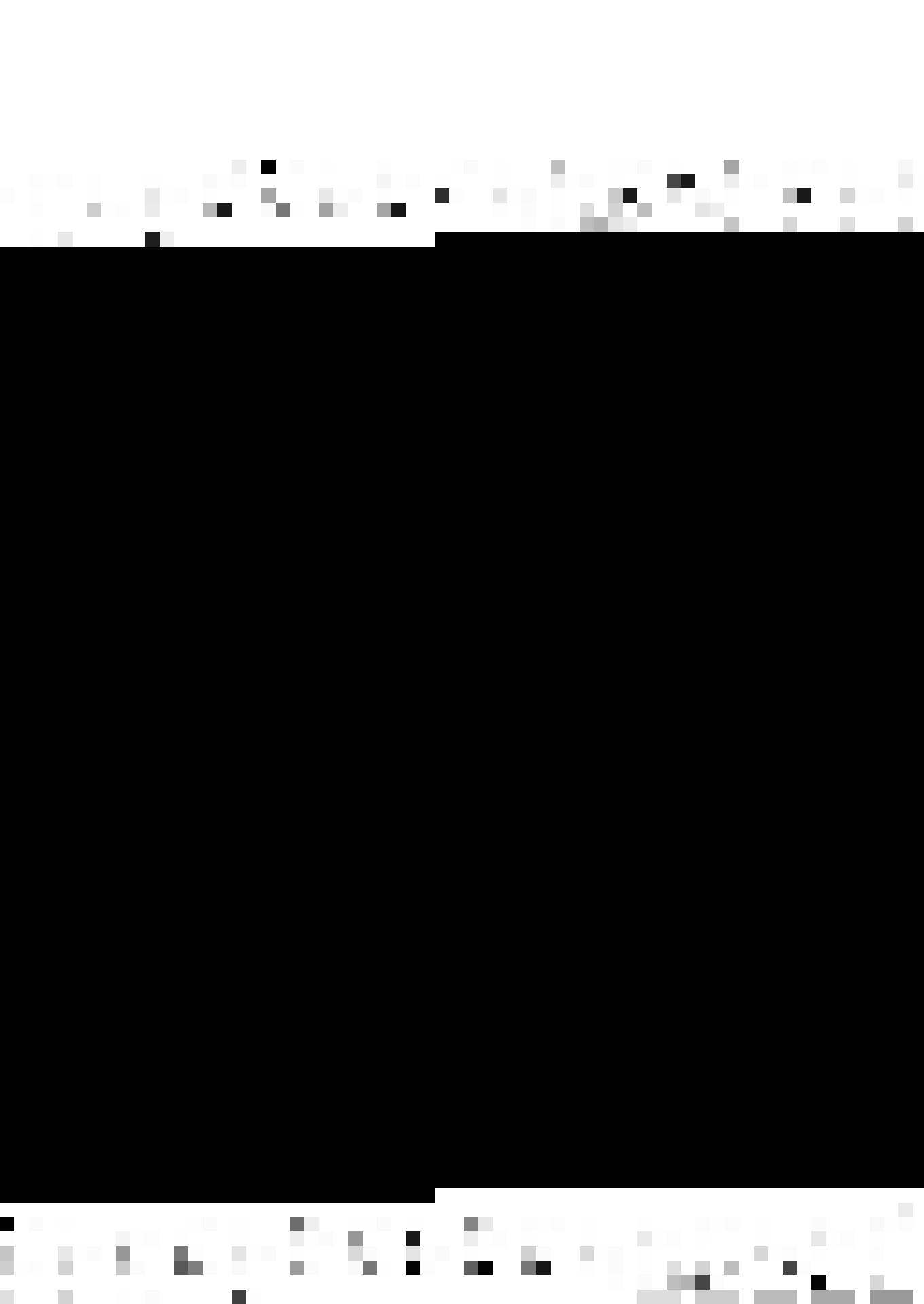
1.  $S \rightarrow ((AB, [2, 1]), BA, (AB, [2, 1]))$ ,
2.  $A \rightarrow ((AaBAb, [2, 3, 1]), (aaABA, [2, 1, 3]), (AAabbB, [3, 2, 1]))$ ,
3.  $A \rightarrow (a, a, \epsilon)$ , and
4.  $B \rightarrow (b, \epsilon, \epsilon)$ .

Observe that  $G$  is an  $(3, \{3\}, \emptyset, 3)CG$ .

Now,  $G' = (V_N, V_T, P', (S)^3)$ , where  $P'$  comprises of the productions:

1.  $S \rightarrow (AB, (BA, [2, 1])), AB)$ ,
2.  $A \rightarrow (AaBAb, (aaABA, [3, 2, 1]), (AAabbB, [1, 3, 2]))$ ,
3.  $A \rightarrow (a, a, \epsilon)$ , and
4.  $B \rightarrow (b, \epsilon, \epsilon)$ .

Again, observe that  $G'$  is an  $(3, \{3\}, \emptyset, 3)CG$ . Also, clearly for  $w_i$ ,  $1 \leq i \leq 3$ , in  $V_T^*$ ,  $(S)^3 \xrightarrow[G]{*} (w_1, w_2, w_3)$ , iff  $(S)^3 \xrightarrow[G']{*} (w_1, w_2, w_3)$ .



We now give the following convention:

Convention 1.2.1:

Unless otherwise specified, it shall always be understood that all nonterminating productions in an  $(n, \sigma, k)$  CG are of the type:  $A \rightarrow (\alpha_1, (\alpha_2, \kappa_2), \dots, (\alpha_n, \kappa_n))$ .

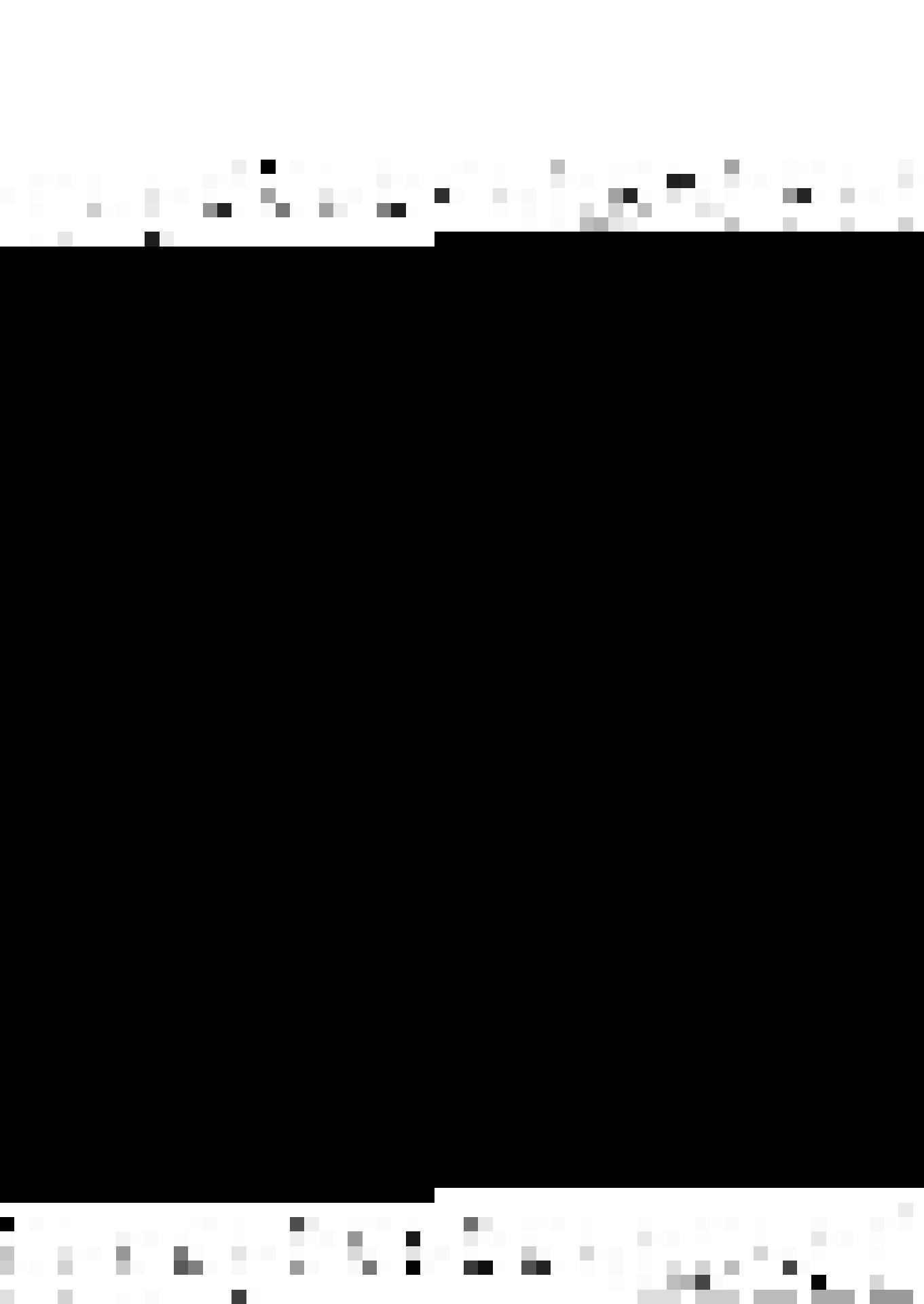
1.3  $n$ -DERIVATION TREES<sup>1</sup>:

We know that the concept of a derivation tree plays an important role in the theory of context-free languages. Can we have a similar concept for coupled languages and coupled translations? In this section, we shall discuss a similar concept:  $n$ -derivation tree,  $n \geq 1$ , which, as we shall see in the next chapter, plays an important role in studying the structure of coupled languages.

$n$ -derivation trees will be representations of derivations in  $n$ -coupled grammars<sup>2</sup>, similarly as derivation trees are representations of derivations in context-free grammars.

Now, we introduce the concept of an  $n$ -derivation tree and some related concepts, through the following recursive definition:

- 
1. Concepts about trees, which are not defined here can be taken from (18).
  2. While defining an  $n$ -derivation tree in an  $n$ -coupled grammar  $G$ , we will not assume that  $G$  is such that the permutations in the first components of the forms which comprise the right hand sides of the nonterminating productions in  $P$ , are necessarily identity permutations. This is just a matter of



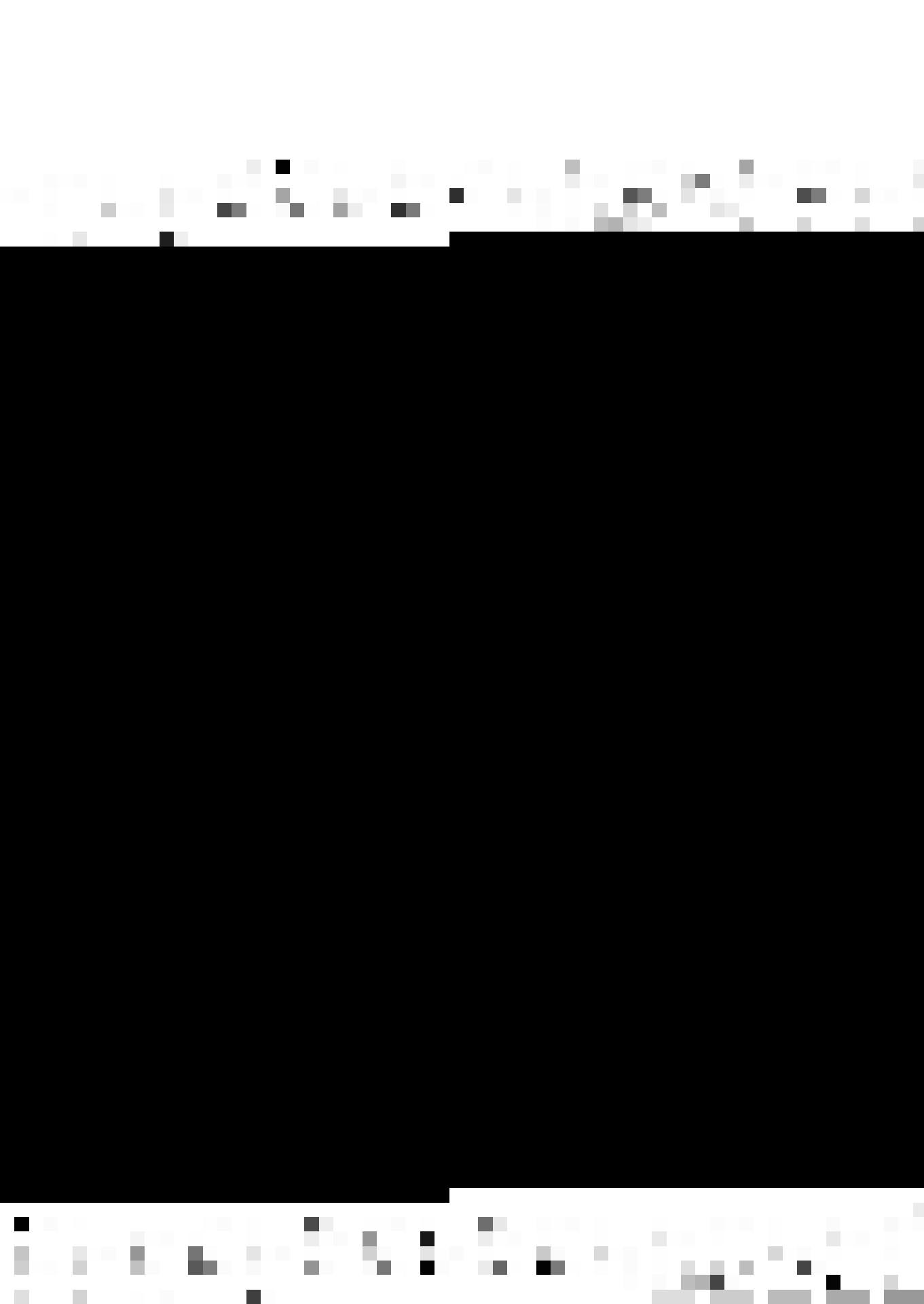
Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, \sigma, k)CG$ . An  $n$ -tuple  $T = \langle T_1, \dots, T_n \rangle$  of labeled rooted trees with permutations associated with some<sup>1</sup> nonpendant nodes in  $T_i$ , is an  $n$ -derivation tree in  $G$  if:

1. The label of every nonpendant node is in  $V_N$  and that of every pendant node is in  $V_T \cup \{\epsilon\}$ . Every node in  $T_i$ , having  $m (\geq 1)$  nonpendant direct descendants, is associated with a permutation on  $m$  objects.
2. The root  $R_i$  of each tree  $T_i$  has label  $S$ . The  $n$ -tuple:  $\langle R_1, \dots, R_n \rangle$ , is called an  $n$ -node of  $T$  and is said to have label  $S$ . It is also called the  $n$ -root of  $T$ .
3. Let  $\langle N_1, \dots, N_n \rangle$  be an  $n$ -node in  $T$  with label  $A$ . Let  $\pi_i$  be the permutation associated with  $N_i$ . Let  $A_{i1}, \dots, A_{im_i}$  be the labels of the direct descendants of  $N_i$  in order from the left. Then  $A \rightarrow \alpha$ , is a production in  $P$ , where

$$\alpha = ((A_{11} \dots A_{1m_1}, \pi_1), \dots \dots \dots, (A_{n1} \dots A_{nm_n}, \pi_n))$$

If  $M_i$  is a direct descendant of  $N_i$  in  $T_i$ ,  $1 \leq i \leq n$ , such that the labels of  $M_i$ ,  $1 \leq i \leq n$ , correspond with each other in the form  $\alpha$  (in  $G$ ), then the  $n$ -tuple:  $\langle M_1, \dots, M_n \rangle$  is called an  $n$ -node. Clearly, each  $M_i$  has the same label, this common label is said to be the label of the  $n$ -node:  $\langle M_1, \dots, M_n \rangle$ .

1. Actually all those nodes which have nonpendant descendants.



4. Let  $a_{i1}, \dots, a_{im_i}$  be the labels (all in  $V_T \cup \{\epsilon\}$ ) of the direct descendants of  $N_{i1}$ , in order from the left. Then,

$A \rightarrow (a_{11} \dots a_{1m_1}, \dots, a_{n1} \dots a_{nm_n})$ , is in P.

\*\*\*

When we say that T is an n-derivation tree, it will be taken to mean that it is an n-derivation tree in some  $(n, \sigma, k)CG$ .

We observe that if  $T = \langle T_1, \dots, T_n \rangle$  is an n-derivation tree<sup>1</sup> in an  $(n, \sigma, k)CG$  in which each nonterminating production is of the type:  $A \rightarrow (\alpha_1, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n))$ , then all permutations associated with nodes in  $T_1$  will be identity permutations. (Note in this connection Convention 1.2.1).

The following are the definitions of certain concepts related to the concept of n-derivation trees.

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an n-derivation tree in an  $(n, \sigma, k)CG$ .

Let  $p_i = N_{i1}, \dots, N_{im}$  be a path in  $T_i$  such that  $N_{i1}$  is the root of  $T_i$  and  $N_{im}$  is the 'farthest' nonpendant node from  $N_{i1}$  (on this path) and for all  $s$ ,  $1 \leq s \leq m$ ,  $M_s = \langle N_{1s}, \dots, N_{ns} \rangle$ , is an n-node in T. Then the n-tuple:  $\langle p_1, \dots, p_n \rangle$ , is called an n-path in T; the paths  $p_1, \dots, p_n$  are said to correspond with each other. The same n-path may also be

1. If there is no permutation associated with a node in  $T_i$  which has got nonpendant direct descendants, then it will be understood that the permutation associated with it is an identity permutation.



represented as a sequence of n-nodes:  $M_1, \dots, M_n$ . (The length of this n-path will be  $n-1$ ):

Let  $P_1 = \langle p_{11}, \dots, p_{1n} \rangle$ , and  $P_2 = \langle p_{21}, \dots, p_{2n} \rangle$ , be n-paths in  $\mathbb{P}$ .  $P_1$  is said to be to the left of  $P_2$  iff  $p_{11}$  is to the left of  $p_{21}$  in  $P_1$ .

A subtree of  $T$  is the n-tuple:  $T^S = \langle T_1^S, \dots, T_n^S \rangle$ , such that (i)  $R^S = \langle R_1^S, \dots, R_n^S \rangle$ , is an n-node in  $T$ . (ii)  $T_i^S$ ,  $1 \leq i \leq n$ , is a subtree<sup>1</sup> of  $T_i$  rooted at  $R_i^S$ .  $R^S$  is called the n-root of  $T^S$ .

Let  $\alpha_i^S$  be the result of  $T_i^S$ , then  $(\alpha_1^S, \dots, \alpha_n^S)$  is said to be the n-result of  $T^S$  and  $\alpha^S = \alpha_1^S \dots \alpha_n^S$  is said to be the result of  $T^S$ .

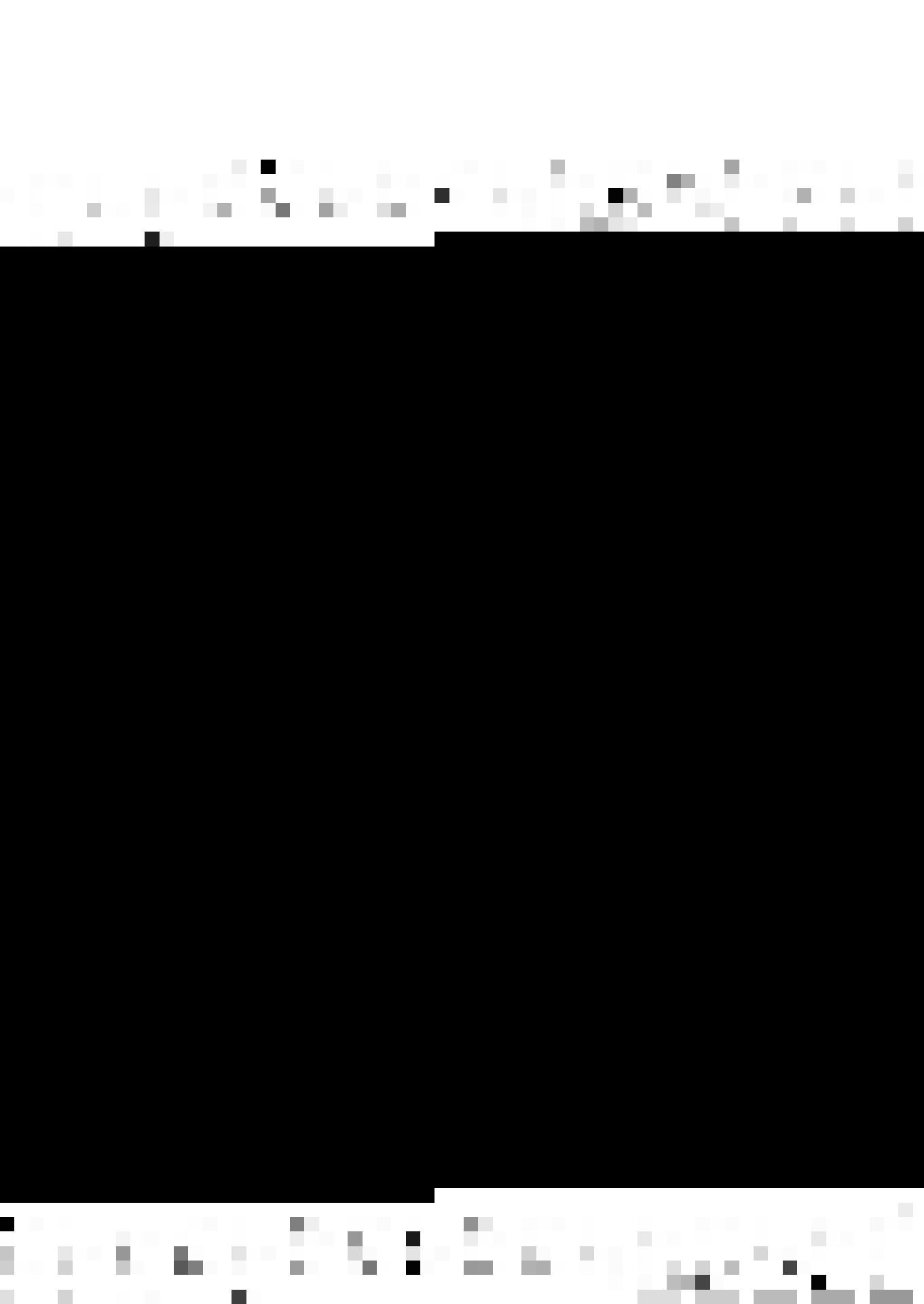
Let us illustrate the concepts introduced above, through an example.

#### Example 1.3.1:

Let  $G = (\{S, A\}, \{a, b, c\}, P, (S)^3)$ , be a  $(3, \emptyset, \{3\}, 2)CG$ , where  $P$  comprises of the productions:

1.  $S \rightarrow (AcAc, (AcAc, [2, 1]), (AcAc, [2, 1])),$
2.  $A \rightarrow (aA, aA, aA),$

1. The subtree of  $T_i$  rooted at  $R_i^S$  is the node  $R_i^S$  together with all its descendants, the edges connecting them, their labels, and the permutations associated with them (if any).

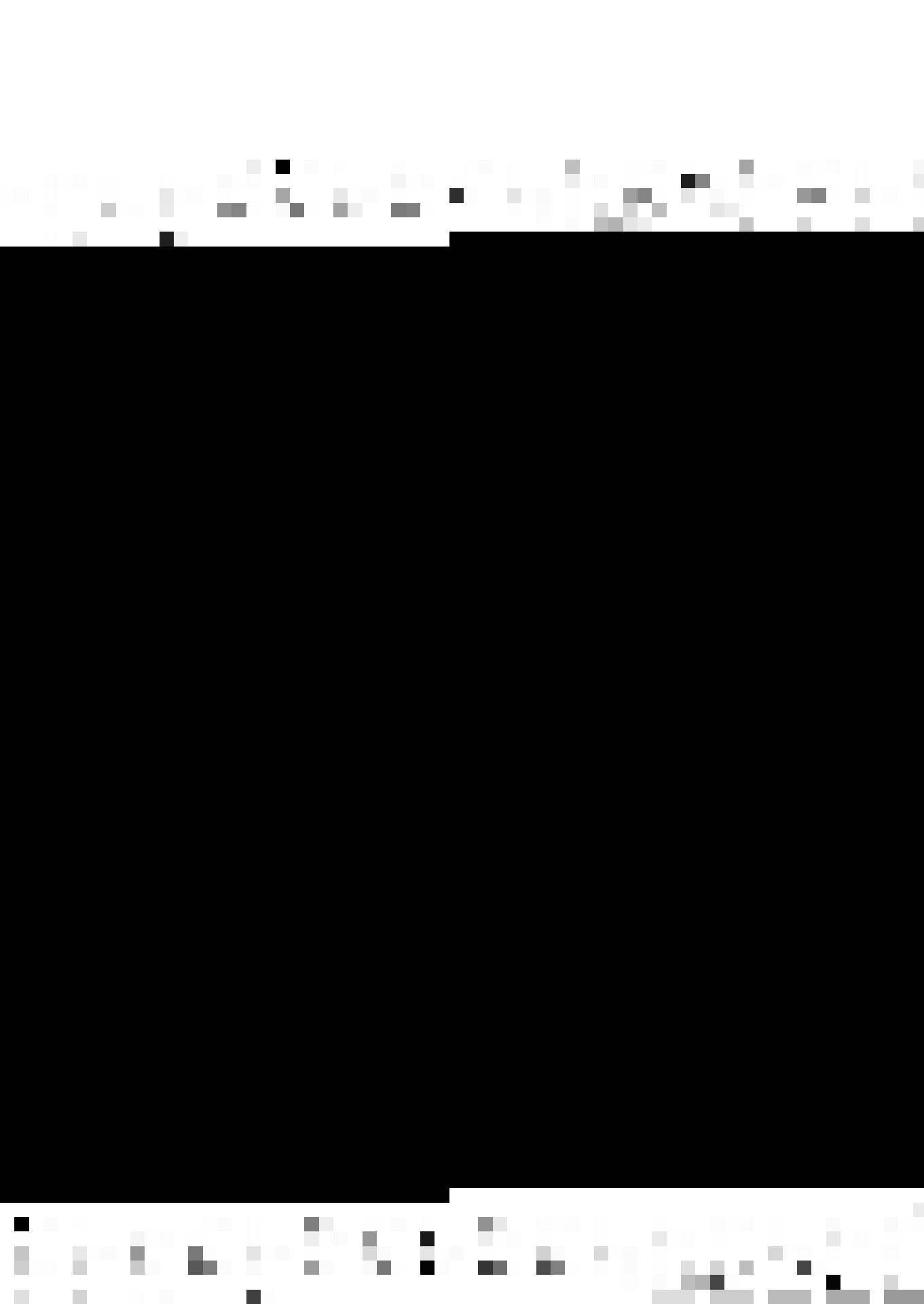


- 3.  $A \rightarrow (bA, bA, bA),$
- 4.  $A \rightarrow (a, a, a),$  and
- 5.  $A \rightarrow (b, b, b).$

We shall graphically represent a 3-derivation tree in  $G,$  just this once with circles instead of points for the nodes. The ordering of the trees comprising the 3-derivation tree will be the same as the left to right ordering of the trees in the graphical representation. The nodes will be numbered for the sake of reference. The labels and permutations (if any) will be shown adjacent to the nodes. Edges are assumed to be directed downwards. See Figure 1.3.1.

In the 3-derivation tree shown in Figure 1.3.1,  $\langle 1, 14, 27 \rangle,$   $\langle 4, 15, 28 \rangle,$   $\langle 2, 17, 30 \rangle,$   $\langle 7, 22, 35 \rangle,$   $\langle 11, 25, 38 \rangle,$  and  $\langle 9, 20, 33 \rangle$  are 3-nodes.  $\langle 1, 14, 27 \rangle$  is the 3-root of the 3-derivation tree. Let  $p_{11} = 1, 2, 7, 11,$   $p_{12} = 1, 4, 9,$   $p_{21} = 14, 15, 20,$   $p_{22} = 14, 17,$   $22, 25,$   $p_{31} = 27, 28, 33,$  and  $p_{32} = 27, 30, 35, 38,$  be some of the paths of the three trees. Then, among others,  $\langle p_{11}, p_{22}, p_{32} \rangle,$  and  $\langle p_{12}, p_{21}, p_{31} \rangle,$  are 3-paths in the 3-derivation tree.  $p_{11}$  corresponds with  $p_{22},$  and with  $p_{32}.$  Similarly,  $p_{12}$  corresponds with  $p_{21},$  and with  $p_{31}.$  The 3-path  $\langle p_{11}, p_{22}, p_{32} \rangle$  can also be represented as the sequence of 3-nodes:

$\langle 1, 14, 27 \rangle, \langle 2, 17, 30 \rangle, \langle 7, 22, 35 \rangle, \langle 11, 25, 38 \rangle .$



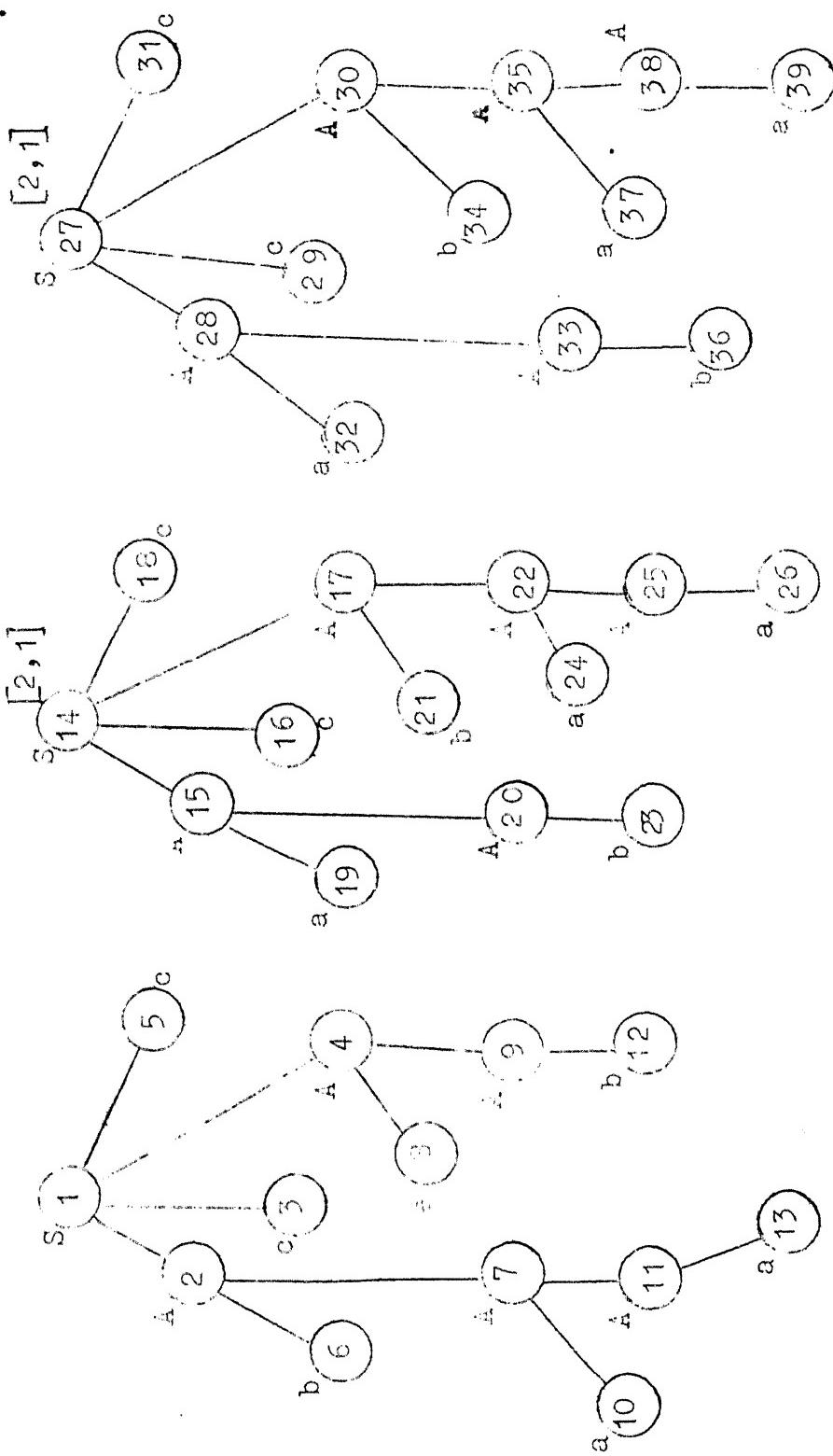
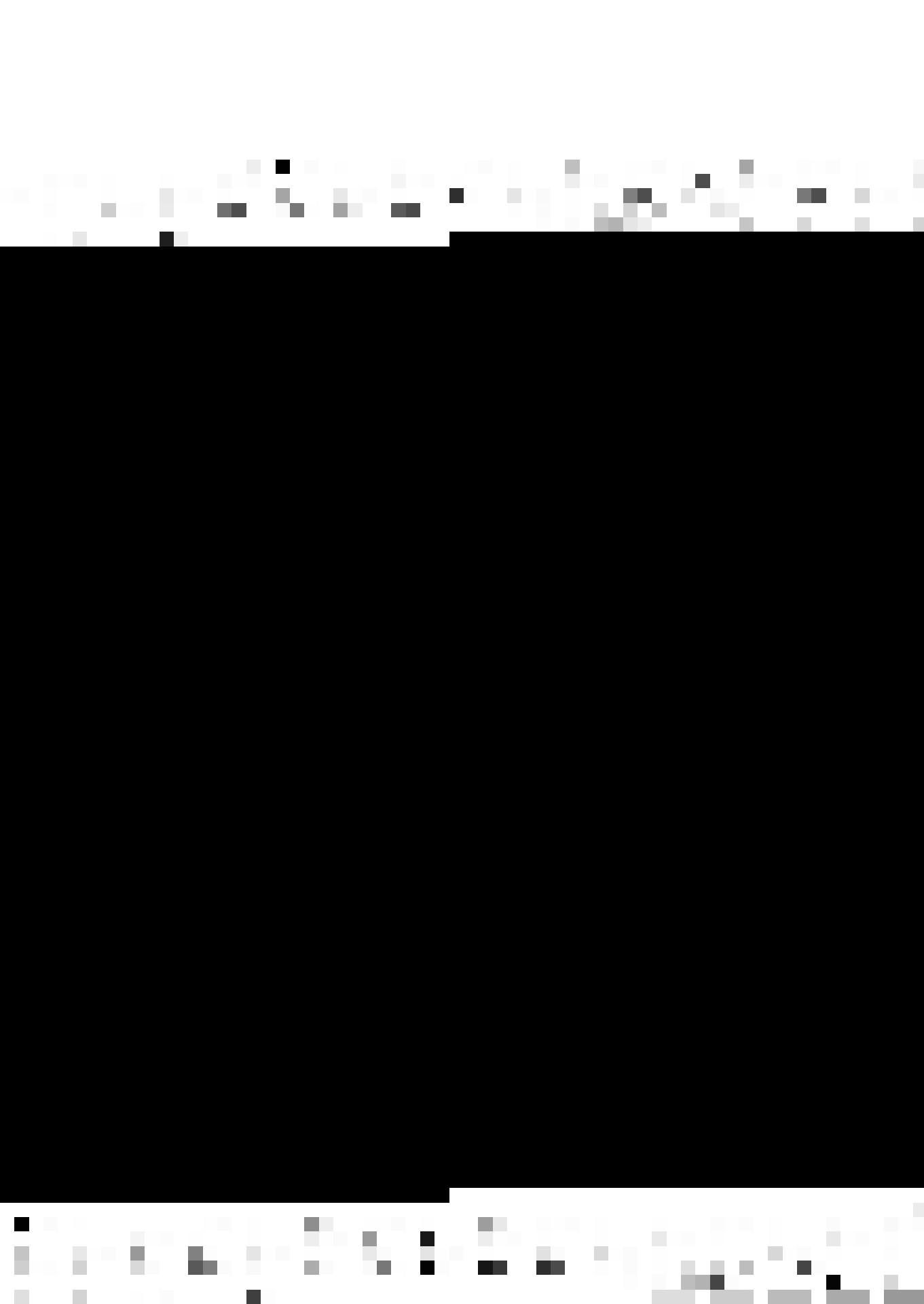


Figure 1.3.1



Similarly, the 3-path  $\langle p_{12}, p_{21}, p_{31} \rangle$ , can also be represented as:

$\langle 1, 14, 27 \rangle, \langle 4, 15, 28 \rangle, \langle 9, 20, 33 \rangle$ .

The label of the 3-node:  $\langle 1, 14, 27 \rangle$ , is S. Similarly, the label of the 3-node:  $\langle 7, 22, 35 \rangle$ , is A.

The 3-path:  $\langle p_{11}, p_{22}, p_{32} \rangle$ , is to the left of the 3-path:  $\langle p_{12}, p_{21}, p_{31} \rangle$ .

The 3-result and the result of the 3-derivation tree are (baaacabc, abcbaac, abcbaac), and (baaacabcaabcbaacabcaac), respectively.

Figure 1.3.2 represents a subtree of the 3-derivation tree with the 3-node:  $\langle 2, 17, 30 \rangle$ , as its 3-root.

\*\*\*

We now give a lemma which establishes the relationship between the derivations and the n-derivation trees in an n-coupled grammar. Its proof is on the same lines as that of the corresponding result for context-free grammars (18), and we omit it.

#### Lemma 1.3.1:

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, \sigma, k)CG$ . Then,  $(S)^n \xrightarrow[G]{*} (w_1, \dots, w_n)$ ,  $w_i$  is in  $V_T^*$ , iff there is an n-derivation tree  $T = \langle T_1, \dots, T_n \rangle$ , in  $G$  with n-result  $(w_1, \dots, w_n)$ . ( $T$  is said to be an n-derivation tree of the sentence  $w = w_1 \dots w_n$  in  $G$ ).



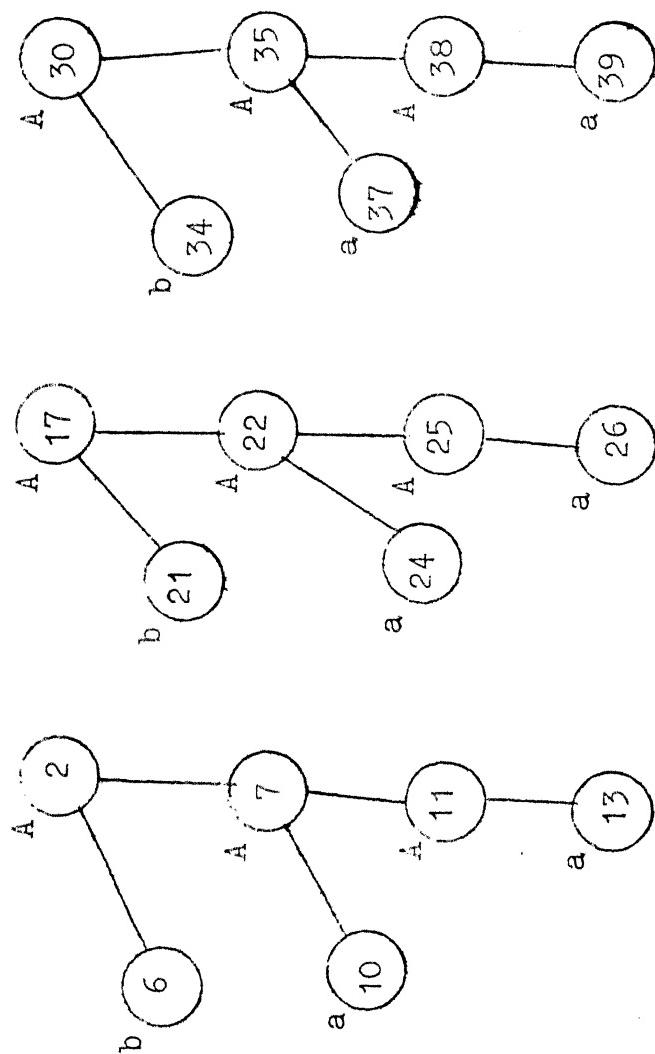


Figure 1.3.2



#### 1.4 SIMPLIFICATION OF COUPLED GRAMMARS:

In this section, we prove only those simplification results for coupled grammars, which will be useful in later chapters. Thus, this section does not exhaust all possible simplifications of coupled grammars. Essentially we prove, here, the 'normal form' theorems for coupled and simple coupled grammars.

We know that any context-free language can be generated by a context-free grammar in Chomsky Normal Form (8). Can we have an analogous result for coupled languages? We will see that this is true for simple coupled languages only and not true, in general, for coupled languages. We will in fact see in the next chapter that 'order' of coupled grammars gives rise to a hierarchy. The following result, however, does hold for coupled languages and coupled translations.

##### Lemma 1.4.1:

Any  $(n, \sigma, k)CL$  (or  $(m, n-m, \sigma, k)CT$ ) can be generated by an  $(n, \sigma, k')CG$   $G = (V_N, V_T, P, (S)^n)$ , where  $k' = k$  if  $k \geq 2$ ,  $k' \leq 2$  if  $k \leq 1$ , and the productions in  $P$  are only of the type:  $A \rightarrow (\beta_1, (\beta_2, \pi_2), \dots, (\beta_n, \pi_n))$  or  $A \rightarrow (b_1, \dots, b_n)$ ,  $\beta_i$ , and  $b_i$  in  $V_N^* - \{\epsilon\}$ , and  $V_T \cup \{\epsilon\}$ , respectively.



Proof:

It suffices to prove that given an  $(n, \sigma, k)$ CG  $G' = (V'_N, V_T, P', (S)^n)$ , we can construct an  $(n, \sigma, k')$ CG  $G = (V_N, V_T, P, (S)^n)$ , which satisfies the conditions specified in the lemma such that for  $w_i$  in  $V_T^*$ ,  $(S)^n \xrightarrow{G'} (w_1, \dots, w_n)$  iff  $(S)^n \xrightarrow{G} (w_1, \dots, w_n)$ .

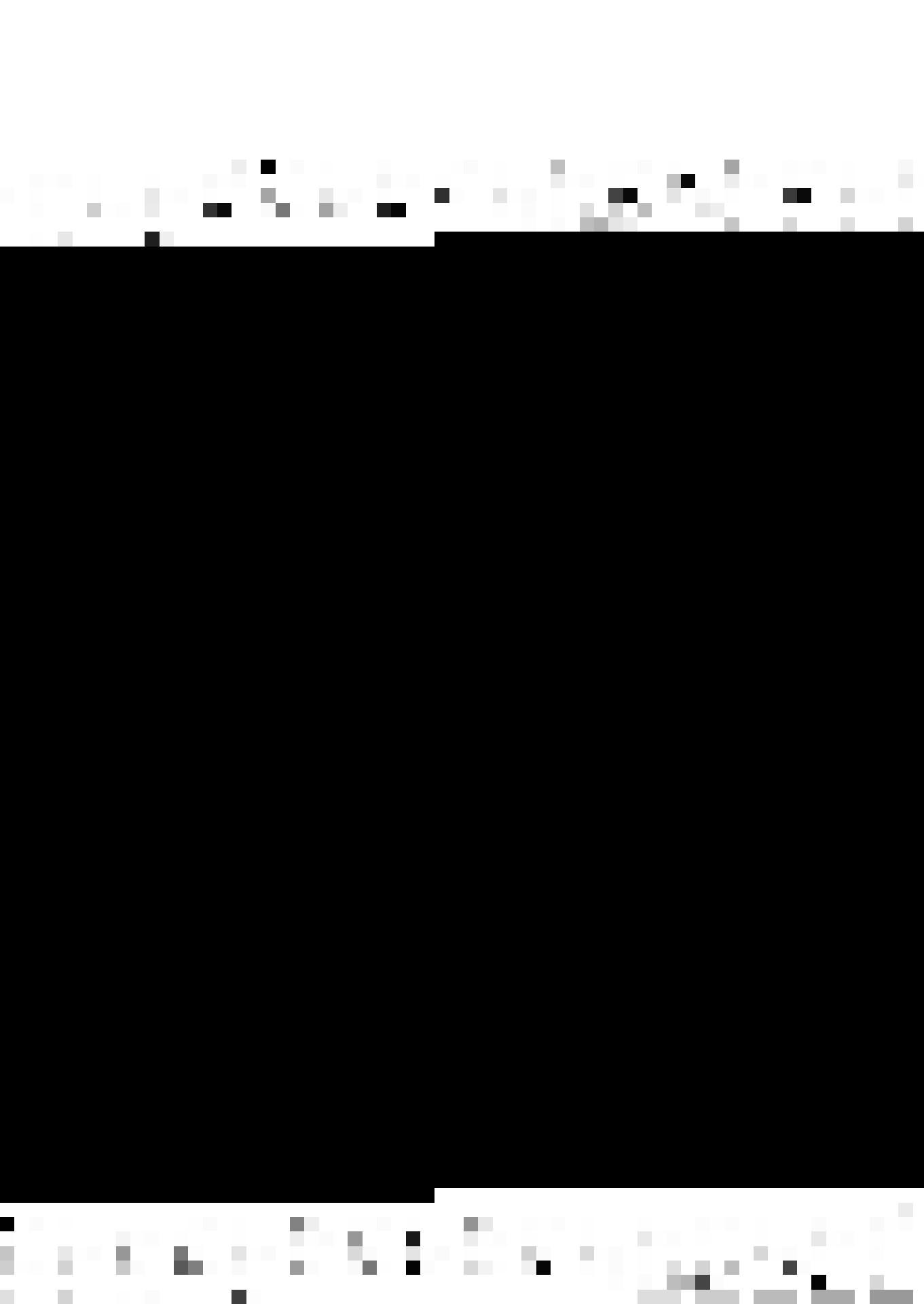
We construct  $G$  from  $G'$  as follows:

To start with,  $V_N = V'_N$  and  $P = \emptyset$ .

For each nonterminating production

$\gamma: A \rightarrow (w_{11}A_1w_{12} \dots A_m w_{1(m+1)}, (w_{21}A\hat{\kappa}_2(1)w_{22} \dots A\hat{\kappa}_2(m)w_{2(m+1)}, \kappa_2), \dots, (w_{n1}A\hat{\kappa}_n(1)w_{n2} \dots A\hat{\kappa}_n(m)w_{n(m+1)}, \kappa_n))$  in  $P'$ , where  $A_i$  is in  $V'_N$  and  $w_{ij}$  is in  $V_T^*$ , we add to  $V_N$ , new nonterminals  $C_1, \dots, C_{m+1}$  and  $D_1, \dots, D_{m+1}$  and add to  $P$  the following productions:

1.  $A \rightarrow (C_{m+1}D_{m+1})^n,$
2.  $D_{m+1} \rightarrow (w_{1(m+1)}, \dots, w_{n(m+1)}),$
3.  $C_{m+1} \rightarrow (C_1 \dots C_m, (C\hat{\kappa}_2(1) \dots C\hat{\kappa}_2(m), \kappa_2), \dots, (C\hat{\kappa}_n(1) \dots C\hat{\kappa}_n(m), \kappa_n)),$
4.  $C_j \rightarrow (D_j A_j)^n, 1 \leq j \leq m, \text{ and}$
5.  $D_j \rightarrow (w_{1j}, w_{2\kappa_2(j)}, \dots, w_{n\kappa_n(j)}), 1 \leq j \leq m.$



(Note that since  $C_1, \dots, C_{m+1}$  and  $D_1, \dots, D_{m+1}$  occur only in the productions introduced above, these productions can only produce the effect of the production  $\lambda$ ).

For each terminating production  $\delta: A \rightarrow (w_1, \dots, w_n)$  in  $P'$ , we add to  $V_N$ , new nonterminals  $E_1, \dots, E_p$ , and  $F_1, \dots, F_{p-2}$ , where  $p = \max\{l_1, \dots, l_n\}$ ,  $l_i = |w_i|$ , and add to  $P$  the productions:

$A \rightarrow (E_1 F_1)^n, F_i \rightarrow (E_{i+1} F_{i+1})^{l_i}, 1 \leq i \leq p-3, F_{p-2} \rightarrow (E_{p-1} E_p)$  and  $E_i \rightarrow (b_{i1}, \dots, b_{in}), 1 \leq i \leq p$ , where  $b_{ij}$  is given by the following rule: For those  $j$ ,  $1 \leq j \leq n$ , such that  $|w_j| \geq i$ ,  $b_{ij}$  is the  $i$ -th occurrence of a symbol in  $w_j$ , in order from the left, and for all other  $j$ ,  $b_{ij} = \delta$ .

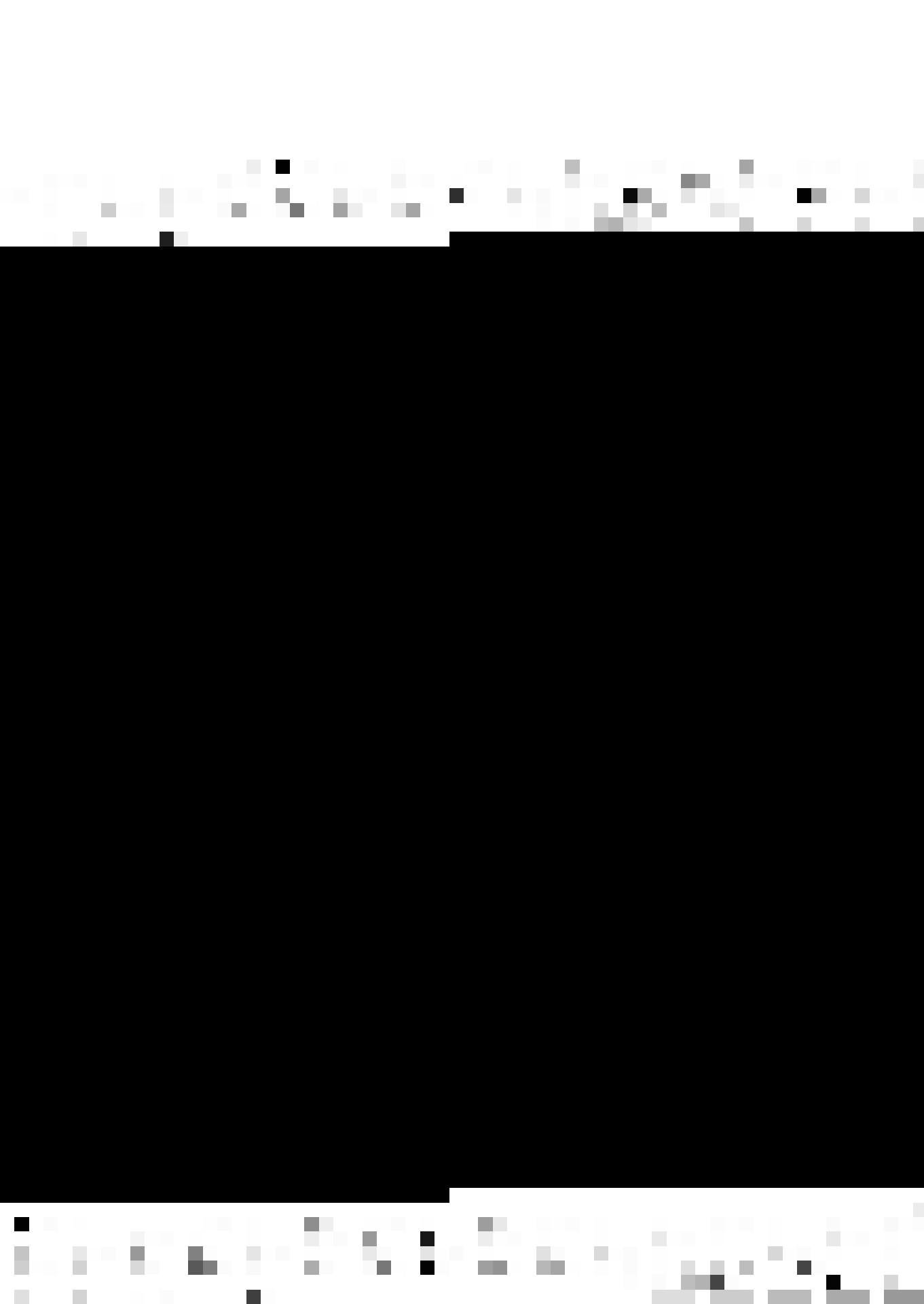
(Again, note that since  $E_1, \dots, E_p$ , and  $F_1, \dots, F_{p-2}$ , occur only in the productions introduced above, these productions can only produce the effect of the production  $\delta$ ).

We omit the rest of the proof, which is straightforward.

Q.E.D.

The following will serve as a useful definition, later:

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, \sigma, k)CG$ . Let  $(S)^n \Rightarrow (\alpha_{11}, (\alpha_{12}, \bar{\alpha}_{12}), \dots, (\alpha_{1n}, \bar{\alpha}_{1n})) \Rightarrow \dots \Rightarrow (\alpha_{m1}, (\alpha_{m2}, \bar{\alpha}_{m2}), \dots, (\alpha_{mn}, \bar{\alpha}_{mn}))$ , be a derivation in  $G$ . Let  $\alpha_{i1} = x_{i1} A_{i1} \beta_{i1}, 1 \leq i \leq m$ , where  $x_{i1}$  is in  $V_T^*$ ,  $\beta_{i1}$  is in  $(V_N \cup V_T)^*$  and  $A_{i1}$  is in  $V_N$ .



The above derivation is a leftmost derivation iff  $(\alpha_{(i+1)1}, (\alpha_{(i+1)2}, \kappa_{(i+1)2}), \dots, (\alpha_{(i+1)n}, \kappa_{(i+1)n}))$  is obtained from  $(\alpha_{i1}, (\alpha_{i2}, \kappa_{i2}), \dots, (\alpha_{in}, \kappa_{in}))$  by expanding  $A_{i1}$ ,  $1 \leq i < m$ .

Based on the above definition, we have the following result which is similar to the corresponding result for context-free languages(18); the proof is also on similar lines and we will omit it.

#### Lemma 1.4.2:

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, \sigma, k)$ CG. For  $w_i$ ,  $1 \leq i \leq n$ , in  $V_T^*$ , if  $(S)^n \xrightarrow{*} (w_1, \dots, w_n)$ , then  $(S)^n \xrightarrow{*} (w_1, \dots, w_n)$ , by a leftmost derivation.

\*\*\*

We now state the 'normal form' definition and the 'normal form theorem'.

An  $(n, \sigma, k)$ CG,  $G = (V_N, V_T, P, (S)^n)$  is said to be in normal form if it satisfies the following conditions:

- (i)  $P$  comprises of productions of the type:  $A \rightarrow (\beta_1, (\beta_2, \kappa_2), \dots, (\beta_n, \kappa_n))$ , or  $A \rightarrow (b_1, \dots, b_n)$ , only, where  $A$  is in  $V_N$ ,  $\beta_i$  is in  $V_N^* - \{\epsilon\}$ , and  $b_i$  is in  $V_T \cup \{\epsilon\}$ ,
- (ii)  $S$  does not appear on the right hand side of any production.



(iii) If  $A \neq S$ , then  $A \rightarrow (\epsilon)^n$ , is not in  $P$ ,

(iv) For no  $A$  and  $B$  in  $V_N$ , is  $A \rightarrow (B)^n$  in  $P$ .

\*\*\*

Note that if  $G$  is an  $(n, \sigma, k)$ CG in normal form, then  $k$  cannot be equal to one.

Theorem 1.4.1 (Normal Form Theorem):

Any  $(n, \sigma, k)$ CL (or  $(m, n-m, \sigma, k)$ CT) can be generated by an  $(n, \sigma, k')$ CG  $G = (V_N, V_T, P, (S)^n)$  in normal form, where  $k' = k$  if  $k \geq 2$  and  $k' \leq 2$  if  $k \leq 1$ .

Proof:

In view of Lemma 1.4.1, it suffices to prove that given an  $(n, \sigma, k)$ CG  $G' = (V'_N, V'_T, F', (S_1)^n)$  satisfying condition (i)

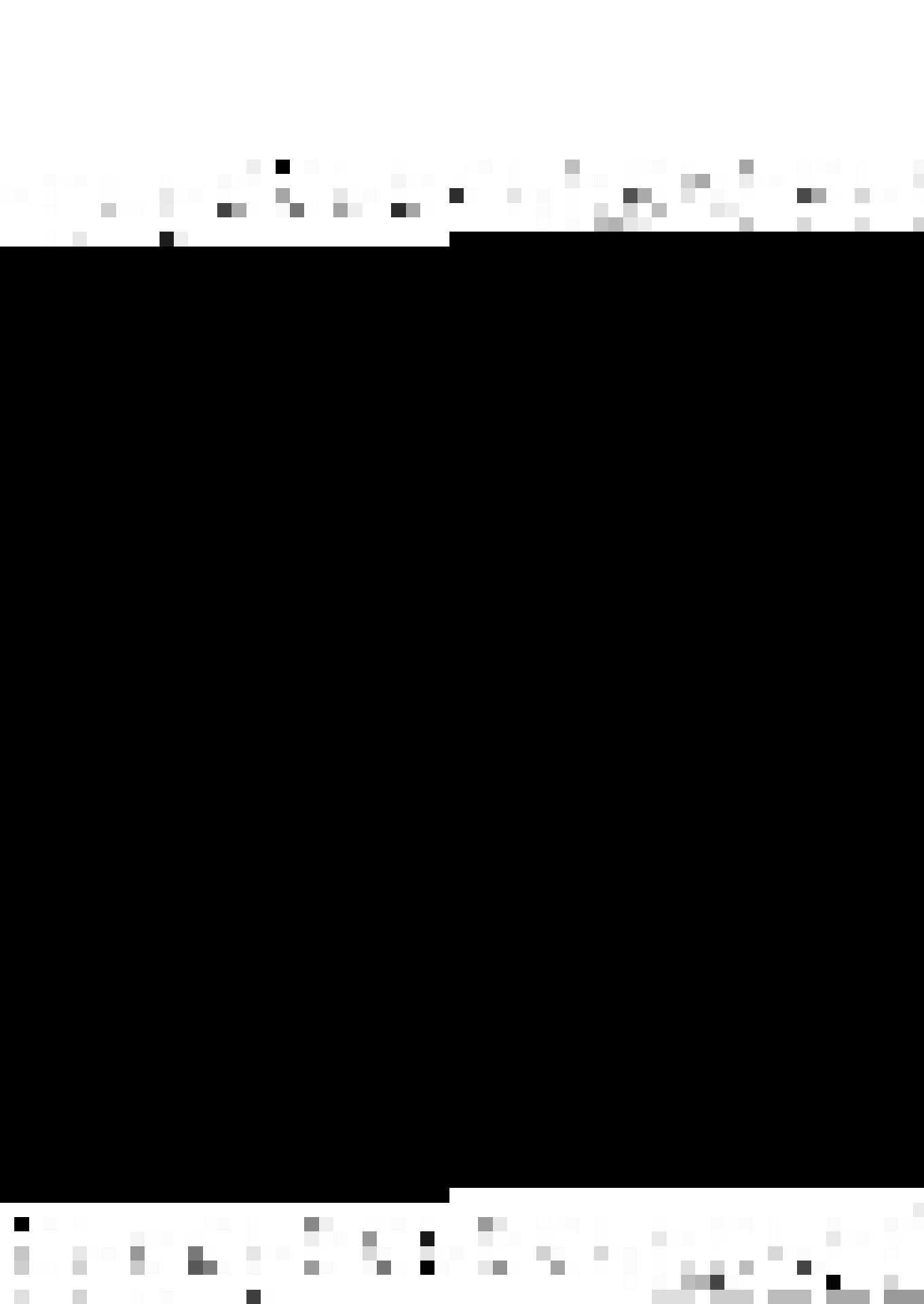
of the normal form definition, we can construct an  $(n, \sigma, k)$ CG  $G = (V_N, V_T, P, (S)^n)$ , in normal form, such that

$$(S)^n \xrightarrow[G]{*} (w_1, \dots, w_n) \text{ iff } (S_1)^n \xrightarrow[G]{*} (w_1, \dots, w_n),$$

where  $w_i$  is in  $V_T^*$ .

This proof is on similar lines as that of the corresponding result for context-free languages (18). We shall, therefore, only give the various constructions and note some crucial points. We shall construct  $G$  from  $G'$  in three parts as follows:

(a) Let  $G' = (V'_N, V'_T, F', (S_1)^n)$  be an  $(n, \sigma, k)$ CG satisfying condition (i) of the normal form definition.



Let  $S$  be a symbol which is not in  $V_N' \cup V_T$ . Let  
 $G = (V_N, V_T, P, (S)^n)$ , where  $V_N = V_N' \cup \{S\}$  and  
 $P = P' \cup \{S \rightarrow \alpha \mid S_1 \rightarrow \alpha, \text{ is in } P'\}$ .

We observe that  $G$  satisfies conditions (i) and (ii) of the normal form definition and is an  $(n, \sigma^-, k)CG$ .

(b) Let  $G' = (V_N', V_T, P', (S_1)^n)$  be an  $(n, \sigma^-, k)CG$  satisfying conditions (i) and (ii) of the normal form definition.

We note that according to Theorem 4.2.3, we can decide whether  $(A)^n \xrightarrow[G]{*} (\epsilon)^n$ , for any  $A$  in  $V_N'$ . If  $A \neq S_1$  and  $(A)^n \xrightarrow[G']{*} (\epsilon)^n$ , then say  $A$  is "type 1". Otherwise, say  $A$  is "type 2".

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $nCG$ , where  $V_N = V_N'$ ,  $S = S_1$ , and  $P$  is constructed from  $P'$  as follows:

1. The terminating productions in  $P$  are the same as those in  $P'$ .

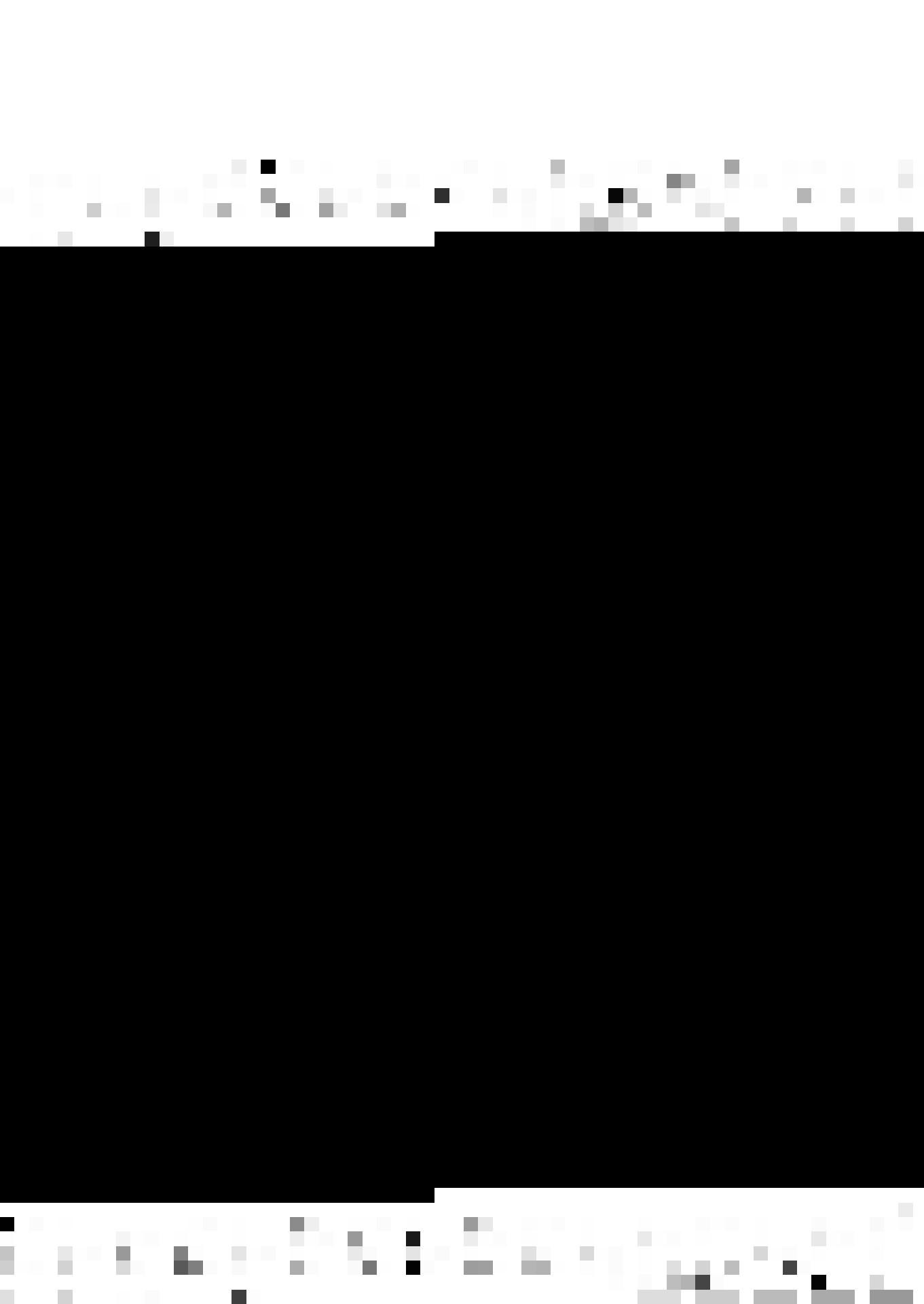
2. If  $(S_1)^n \xrightarrow[G]{*} (\epsilon)^n$ , then  $S \rightarrow (\epsilon)^n$ , is in  $P$ .

3. No other production of the type :  $A \rightarrow (\epsilon)^n$ , is in  $P$ .

4. If  $A \rightarrow (B_1 \dots B_m, (\beta_2, \kappa_2), \dots, (\beta_n, \kappa_n))$ , is a nonterminating production in  $P'$ , then each production of the form:

$$A \rightarrow (\alpha_1 \dots \alpha_m, (\gamma_2, \kappa'_2), \dots, (\gamma_n, \kappa'_n)),$$

is in  $P$ , where for  $i$ ,  $1 \leq i \leq m$ , if  $B_i$  is type 2, then  $\alpha_i = B_i$  and if  $B_i$  is type 1, then  $\alpha_i$  may be  $B_i$  or  $\epsilon$ , and each  $\gamma_i$  is in  $V_N^* - \{\epsilon\}$ .



Note that since  $G'$  is an  $(n, \sigma, k)$ CG satisfying condition (i) of the normal form definition, once the permutations  $\pi_2, \dots, \dots, \pi_n$ , and the string  $B_1 \dots B_m$  are specified,  $(\beta_2, \pi_2), \dots, (\beta_n, \pi_n)$ , get uniquely specified. Similarly, because  $G$  is an nCG and  $\gamma_i, 2 \leq i \leq n$ , is in  $V_N^* - \{\epsilon\}$ ,  $(\gamma_2, \pi'_2), \dots, (\gamma_n, \pi'_n)$ , get uniquely specified from the string  $\alpha_1 \dots \alpha_m$ , and the permutations  $\pi_2, \dots, \pi_n$ .

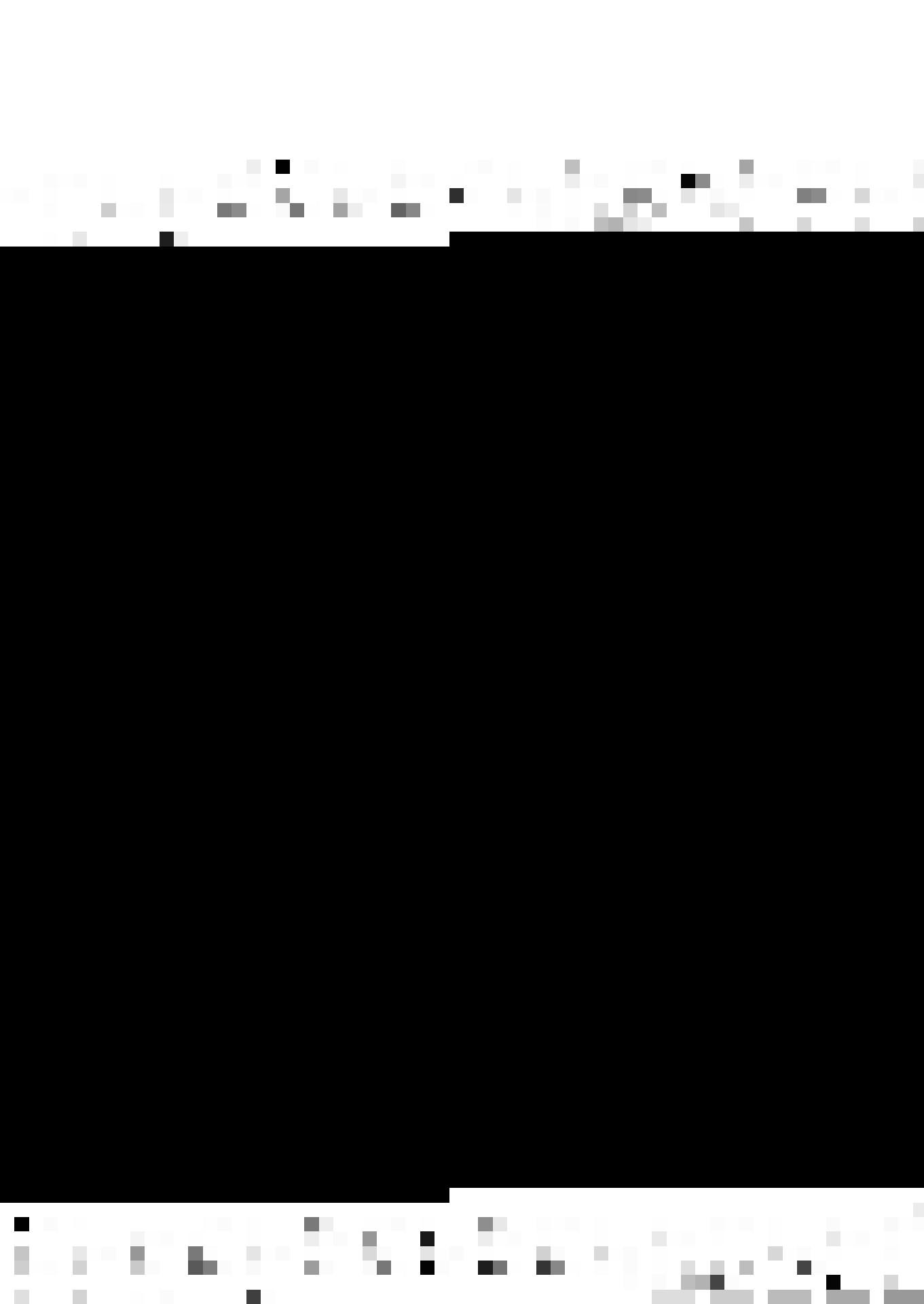
Also note that each nonterminating production in  $P'$  gives rise to itself in  $P$  and may give rise to one or more additional productions and the 'additional' productions in  $P$  do not 'contribute' to the  $\mu$ -simplicity of  $G$  (i.e., the  $\mu$ -simplicity would be the same even if these productions were not present).

Observe that  $G$  satisfies conditions (i), (ii), and (iii) of the normal form definition and is an  $(n, \sigma, k)$ CG.

(c) Let  $G' = (V_N', V_T, P', (S_1)^n)$  be an  $(n, \sigma, k)$ CG satisfying conditions (i), (ii), and (iii) of the normal form definition. Let an nCG  $G = (V_N, V_T, P, (S)^n)$  be constructed from  $G'$  as follows:

$V_N = V_N'$ ,  $S_1 = S$ , and  $P$  is constructed from  $P'$  in the following manner.

Call the productions in  $P'$  of the type  $A \rightarrow (B)^n$ , where  $B$  is in  $V_N'$ , as "type 1" productions. Call all other productions of  $P'$  as "type 2" productions.



First include all type 2 productions of  $P'$  in  $P$ . Add to  $P$  all productions:

$A \rightarrow (\alpha_1, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n))$ , where

$B \rightarrow (\alpha_1, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n))$ , is a type 2 production and  $(A)^n \xrightarrow[G']{*} (B)^n$ .

We observe that we can easily test if  $(A)^n \xrightarrow[G']{*} (B)^n$ , since if  $(A)^n \xrightarrow[G']{*} (B_1)^n \xrightarrow[G']{*} \dots \xrightarrow[G']{*} (B_m)^n \xrightarrow[G']{*} (B)^n$ , and some nonterminals appear twice in the sequence, we can find a shorter sequence which uses type 1 productions only such that  $(A)^n \xrightarrow[G']{*} (B)^n$ . Note that  $(A)^n \xrightarrow[G']{*} (B_1 \dots B_m, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n)) \xrightarrow[G']{*} (B)^n$ , where  $m > 1$ , and  $B_1, \dots, B_m$  are in  $V_N$ , is not possible because  $G'$  satisfies condition (iii) of the normal form definition.

Observe that  $G$  is an  $(n, \sigma, k)CG$  in normal form.

Q.E.D.

Cor. 1.4.1.1:

Any nCL (or  $(m, n-m)CT$ ) can be generated by an nCG  $G$  in normal form.

\*\*\*

We now have the normal form definition and theorem for simple coupled grammars. To distinguish it from the normal form of coupled grammars, we call the normal form of simple coupled grammars as the 'simple normal form'. We will see that the simple normal form for simple coupled grammars is



very much like the 'Chomsky Normal Form' for the context-free grammars (8).

An nSCG  $G = (V_N, V_T, P, (S)^n)$  is said to be in simple normal form if it is in normal form<sup>1</sup> and the nonterminating productions in  $P$  are only of the type:  $A \rightarrow (BC)^n$ .

Theorem 1.4.2:

Any  $(n, k)$  SCI (or  $(m, n-m, k)$  SCT), can be generated by an  $(n, k')$  SCG  $G = (V_N, V_T, P, (S)^n)$ , in simple normal form, where  $k' = 2$  if  $k \geq 2$  and  $k' \leq 2$ , otherwise.

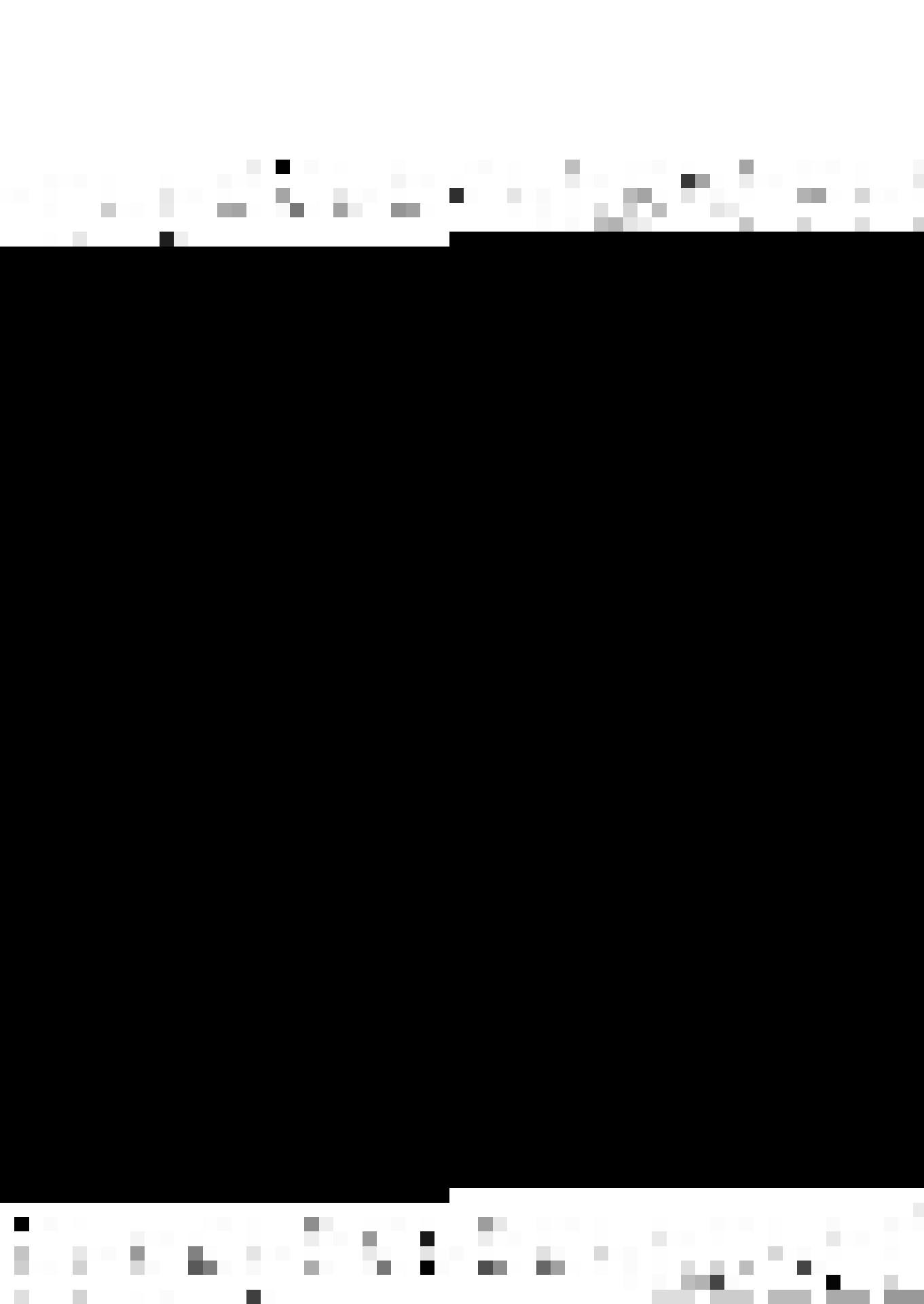
Proof:

In view of Theorem 1.4.1, it suffices to prove that given an  $(n, k)$  SCG,  $k \geq 2$ ,  $G' = (V'_N, V_T, P', (S)^n)$ , in normal form, we can construct an  $(n, 2)$  SCG  $G = (V_N, V_T, P, (S)^n)$ , in simple normal form such that for  $w_i$ ,  $1 \leq i \leq n$ , in  $V_T^*$ ,  $(S)^n \xrightarrow[G]{*} (w_1, \dots, w_n)$ , iff  $(S)^n \xrightarrow[G']{*} (w_1, \dots, w_n)$ .

In order to get  $G$ , we modify  $G'$ , by adding some additional symbols to  $V'_N$  to get  $V_N$  and by replacing some productions of  $P'$  to get  $P$ .

For each nonterminating production:  $A \rightarrow (B_1 \dots B_m)^n$ , in  $P'$ , where  $m \geq 3$ , we add new nonterminals  $D_1, \dots, D_{m-2}$ , and replace  $A \rightarrow (B_1 \dots B_m)^n$ , by the set of productions:

1. Note that every nSCG is an nCG.



$$A \rightarrow (B_1 D_1)^n,$$

$$D_1 \rightarrow (B_2 D_2)^n,$$

:

$$D_{m-3} \rightarrow (B_{m-2} D_{m-2})^n, \text{ and}$$

$$D_{m-2} \rightarrow (B_{m-1} B_m)^n.$$

Note that since the nonterminals  $D_1, \dots, D_{m-2}$  are used only in the above productions, the above productions produce the same effect as the one they replace.

We omit the rest of the proof, which is straightforward.

Q.E.D.

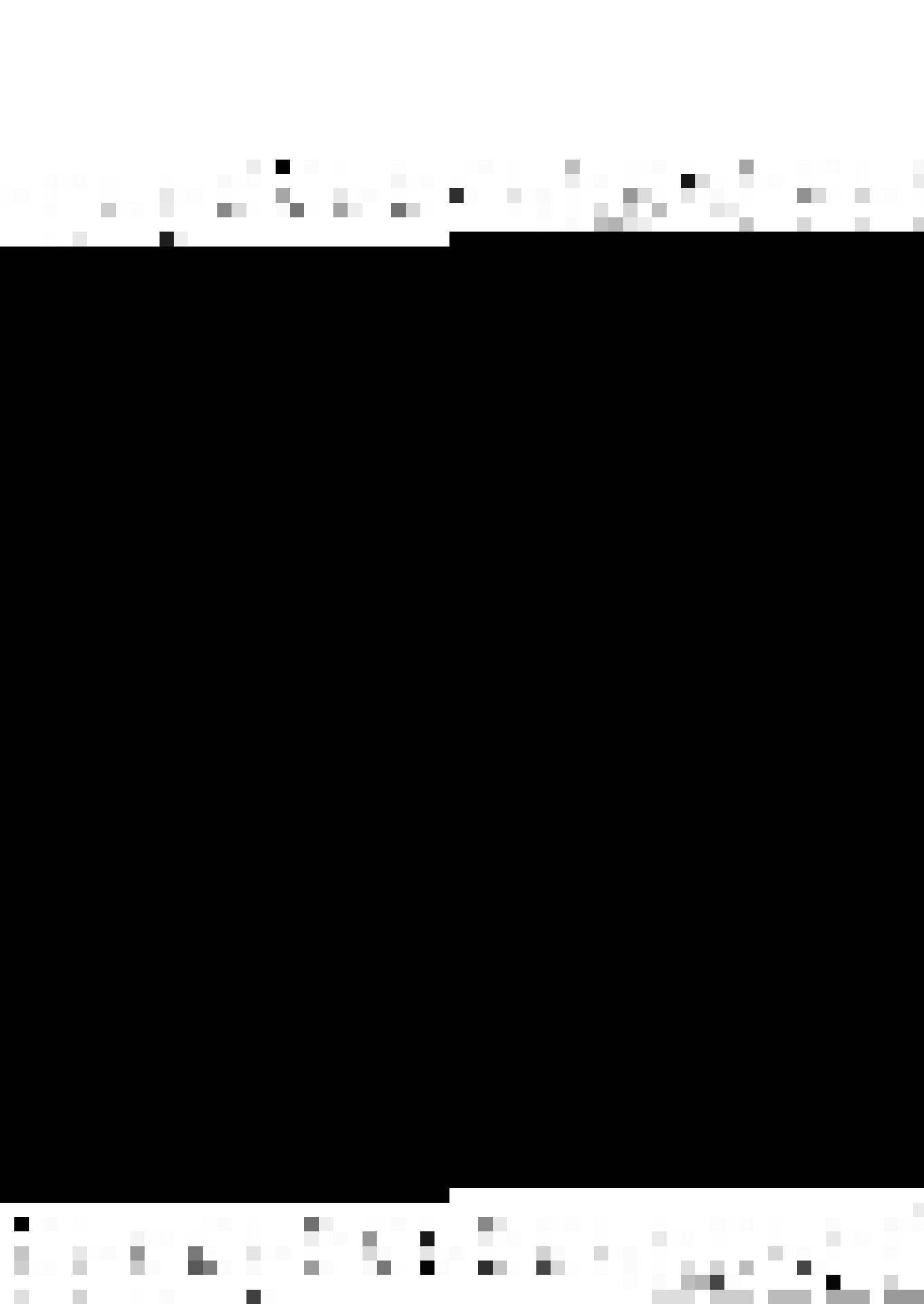
#### Cor. 1.4.2.1:

Any nSCL (or  $(m, n-m)$  SCT) can be generated by an nSCG G in simple normal form.

#### 1.5 CONCLUSIONS:

In this chapter, we have provided the infrastructure on which the thesis is built.

n-coupled grammars, the way derivations are defined in these grammars and the concept of the order of these grammars are generalizations of the corresponding definitions for syntax directed translation schemes (3). In addition to the order, we have defined the degree and the  $\mu$ -simplicity of coupled grammars. All these parameters which are indicators of the



complexity of these grammars induce structure on the languages and the translations they generate, as we shall see in the next chapter. Out of the three parameters, the  $\lambda$ -simplicity is a novel and an interesting concept.

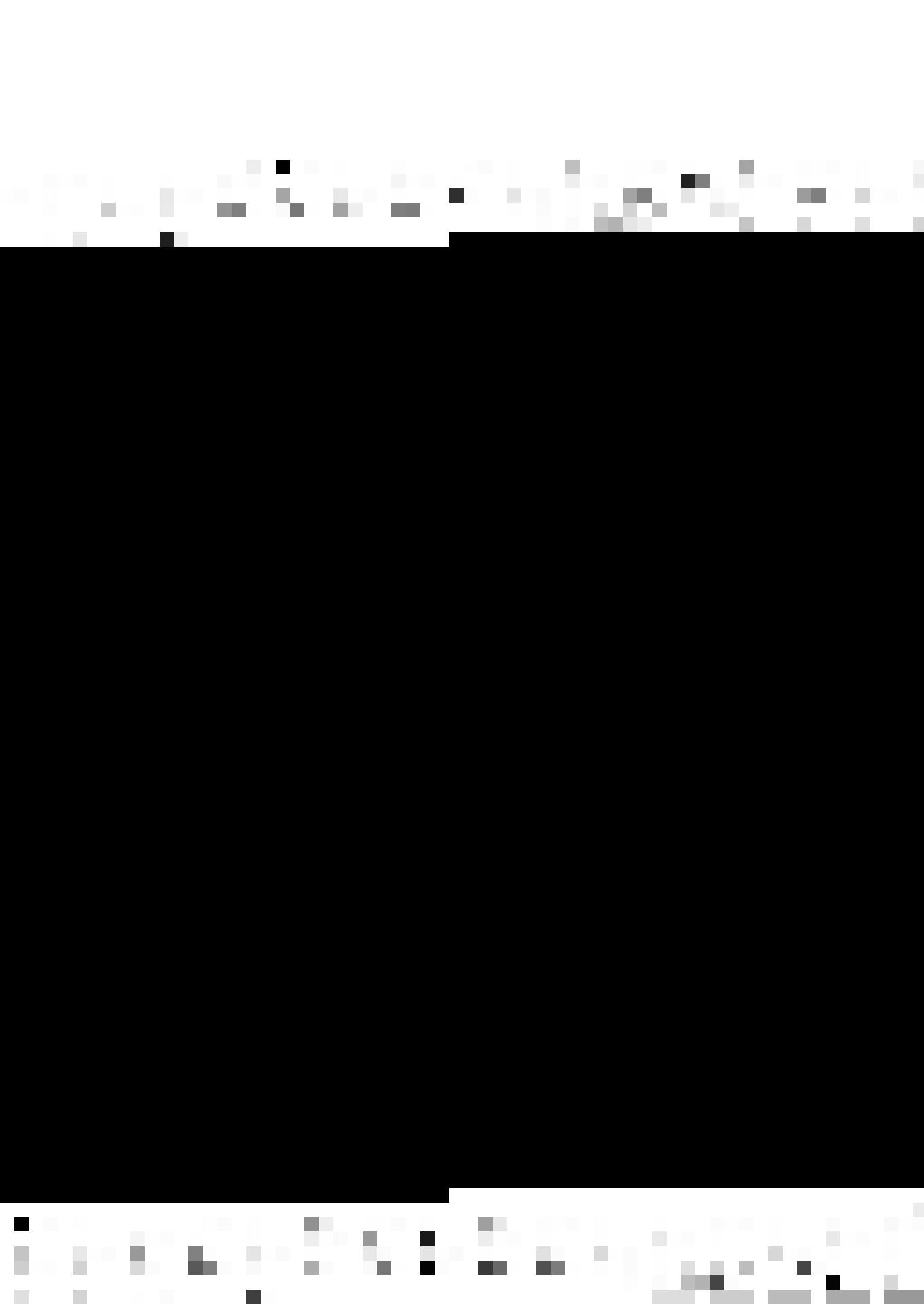
An interesting thing about coupled grammars is that they can generate both languages and translations. We have found out that the domain and range of  $(m, n)$ -coupled translations are  $m$ -coupled languages and  $n$ -coupled languages, respectively. This fact motivates us to concentrate our attention on coupled languages only. We shall see in the next chapter that certain structure results for coupled translations follow as corollaries to the structure results for coupled languages.

$n$ -coupled grammars are very similar to context-free grammars. We have found out that this similarity extends beyond just definitions — it extends to many results as well. This fact will be supported, as we proceed further. This is certainly an encouraging feature of these grammars. Our emphasis in the thesis will not, however, be on this fact, nor is our aim to exhaustively investigate as to which of the results for these grammars differ from those for context-free grammars. We will be concentrating on certain special aspects of these grammars like the structure results, machine characterizations, relationship with some known classes of languages etc., of the languages and/or translations generated by these grammars. We will be, however, continuing to keep a close touch with context-free grammars by pointing out similarities, wherever possible.

In this chapter, we have singled out two subclasses of the class of  $n$ -coupled languages viz., the classes of  $n$ -simple coupled languages and  $n$ -right linear coupled languages. We will find in Chapter 5 that the class of  $n$ -simple coupled languages is the same as the class of simple matrix languages of degree  $n$  (19) and the class of  $n$ -right linear coupled languages is the same as the class of equal matrix languages of order  $n$  (34). This makes us feel that it should be possible to define some restricted types of matrix grammars (1) which will be equivalent to  $n$ -coupled grammars.

The concept of  $n$ -derivation trees and related concepts which we have defined in the chapter play an important role, particularly in the next chapter.

While giving simplification results, we have not dealt with 'unnecessary' nonterminals i.e. those nonterminals which cannot be 'reached' from the start form or those which cannot get reduced to terminals. We, however, conjecture that an nCG (respectively, an nSCG) can be reduced to an equivalent nCG (respectively, an nSCG) which has not got the unnecessary nonterminals. We also conjecture that such a result is not true, in general, for  $(n, \sigma, k)$ CG's.



## CHAPTER 2

### STRUCTURE RESULTS

In this chapter, we investigate how the degree,  $\mu$ -simplicity and order of coupled grammars induce structures on the languages and translations generated by them and how these structures are related with each other.

In section 2.1, we give the basic definitions and results which we use in proving the various results of this chapter. In section 2.2 we mainly concentrate on the structure induced by the degree of coupled grammars. Similarly in sections 2.3 and 2.4, we mainly concentrate on the structure induced by the  $\mu$ -simplicity and order of the coupled grammars, respectively, on the languages and translations generated by them. In these three sections which discuss the structure results (viz. 2.2, 2.3 and 2.4), we also show how these structures are related with each other. Finally, in section 2.5, we give some conclusions.

#### 2.1 BASIC DEFINITIONS AND RESULTS:

In this section, we give the basic definitions and results, which will help us in proving the various structure results later.

The main result of this section is the pumping theorem (Theorem 2.1.1). We will state it in terms of n-derivation



trees; this is part of our strategy for proving various structure results later.

In order to make the proofs less complex, we need to define a number of concepts about labeled trees and n-derivation trees, which we do in this section.

Towards the end of the section, we define the concepts of 'language simplicity predicate' and 'translation simplicity predicate'. These concepts play an important role in proving various structure results. The concept of language simplicity predicate plays a key role in showing that the  $\mathcal{U}$ -simplicity of coupled grammars induces a structure on the languages generated by them.

#### NOTATION:

Let  $T$  be a rooted labeled tree, and let  $M$  be a node in  $T$ . Then, the result<sup>1</sup> of  $T$  is denoted as:  $\text{result}(T)$ , and the result of the subtree of  $T$  rooted at  $M$  is denoted as:  $\text{result}(T^M)$ .

Let  $M$  and  $N$  be two nodes in  $T$  such that  $N$  is a descendant of  $M$ . Clearly, we can write:  $\text{result}(T^M) = x(\text{result}(T^N))y$ , where  $x$  and  $y$  are certain strings (possibly empty). The string  $x$  is then denoted as: l-result( $T^{M-N}$ ) and  $y$  as r-result( $T^{M-N}$ ).

\*\*\*

- 
1. For the definition of the result of rooted labeled trees and other concepts about such trees (not defined here) see (18).



The concepts of  $(M_1, M_2)$ -related n-paths and paths will be used in the proof of the pumping theorem (Theorem 2.1.1) and later also. We define these as follows:

Let  $T$  be a rooted labeled tree (respectively, an n-derivation tree). A path (respectively, an n-path)  $P$  in  $T$  is said to be  $(M_1, M_2)$ -related if  $M_1, M_2$  are nodes (respectively, n-nodes) on  $P$ ,  $M_2$  being a descendant of  $M_1$ , such that the two have got the same labels.

\*\*\*

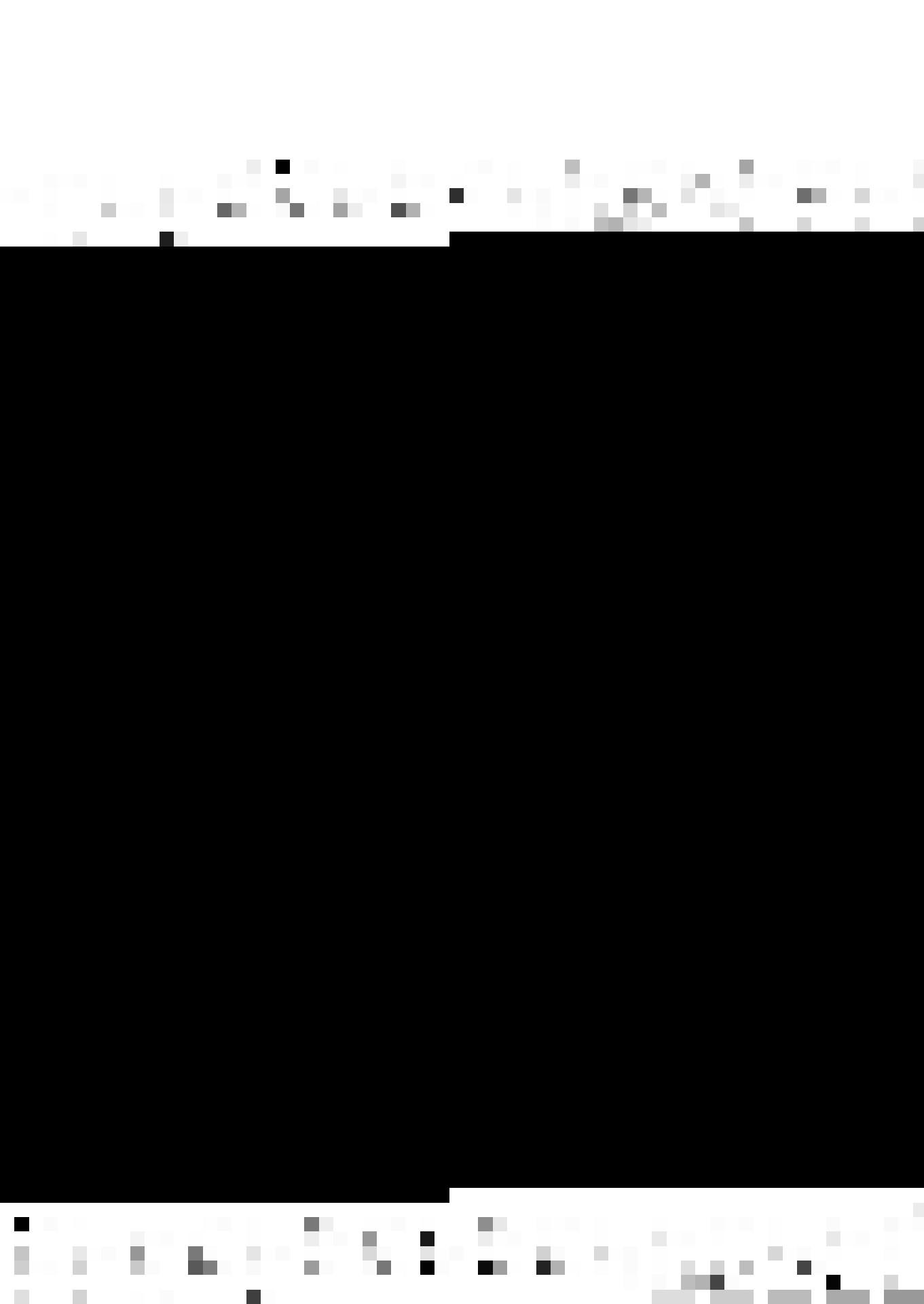
Note that if  $P$  is an  $(M_1, M_2)$ -related path in a derivation tree of a context-free grammar, the labels of  $M_1$  and  $M_2$  will be (the same) nonterminals in the grammar. Again, if  $P$  is an  $(M_1, M_2)$ -related n-path in an n-derivation tree of an n-coupled grammar, the labels of  $M_1$  and  $M_2$  will be the same nonterminals in the grammar.

The following lemma follows from the definition of the n-derivation trees.

#### Lemma 2.1.1:

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an n-derivation tree. Let  $P$  be an n-path in  $T$ , and  $M_1 = \langle M_{11}, \dots, M_{1n} \rangle$ ,  $M_2 = \langle M_{21}, \dots, M_{2n} \rangle$ , two n-nodes on  $P$ , such that for some  $i$ ,  $1 \leq i \leq n$ , there is an  $(M_{1i}, M_{2i})$ -related path in  $T_i$ , then  $P$  is  $(M_1, M_2)$ -related.

\*\*\*



Now, we come to proving the pumping theorem for coupled languages (Theorem 2.1.1). We will first state the following lemma; its proof is simple and is omitted:

Lemma 2.1.2:

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an  $n$ -derivation tree in an  $(n, \sigma, k)CG$ ,  $G$  in normal form. Let  $1 \leq i \leq n$ . Let  $l$  be the maximum number of direct descendants of a node in  $T_i$ , and let there be no path in  $T_i$  of length greater than  $j$ , for some  $j$ ,  $j \geq 1$ . Then,  $|\text{result}(T_i)| \leq l^{j+1}$ .

Theorem 2.1.1:

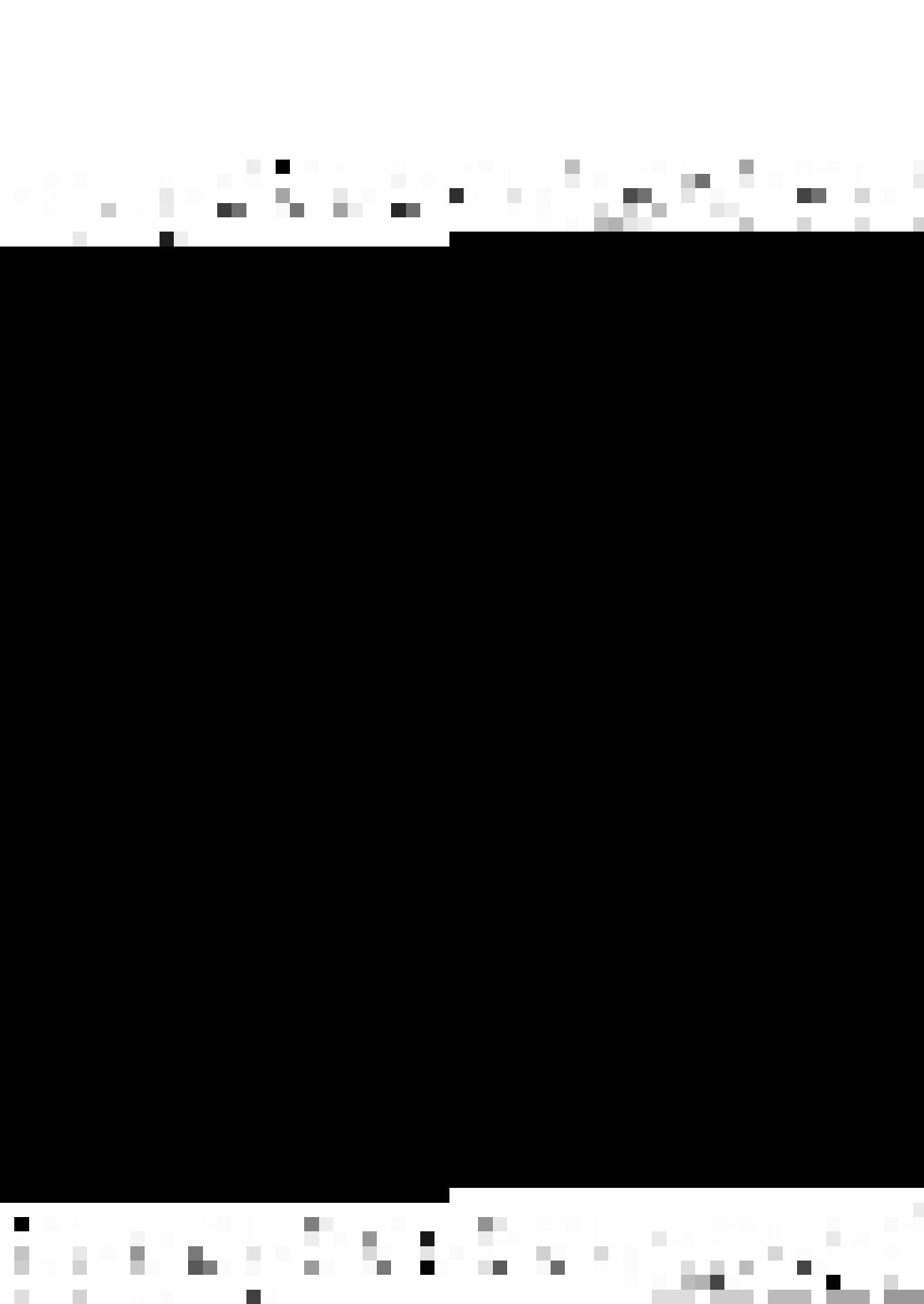
Let  $L$  be an  $(n, \sigma, k')CL$ , defined by an  $(n, \sigma, k)CG$   $G$  in normal form<sup>1</sup>, where  $k = k'$  if  $k' \geq 2$  and  $k \leq 2$ , otherwise. There exist constants  $p$  and  $q$ , depending only on  $L$ , such that if  $z$  is in  $L$ ,  $z_1$  is a substring of  $z$ ,  $|z_1| > n.p$ , and the positions<sup>2</sup> occupied by the symbols comprising  $z_1$  are designated as 'distinguished', then:

(a) there are two  $n$ -nodes  $M_1 = \langle M_{11}, \dots, M_{1n} \rangle$  and  $M_2 = \langle M_{21}, \dots, M_{2n} \rangle$ , in an  $n$ -derivation tree  $T = \langle T_1, \dots, T_n \rangle$  of  $z$  in  $G$ , such that

- (i) there is an  $(M_1, M_2)$ -related  $n$ -path in  $T$ , and
- (ii) for some  $j$ ,  $1 \leq j \leq n$ , either  $l\text{-result}(T_j^{M_1j-M_2j})$  or  $r\text{-result}(T_j^{M_1j-M_2j})$  contains a distinguished position, and for

1. See Theorem 1.4.1.

2. A 'position' in a string of length  $l$  is an integer  $i$  such that  $1 \leq i \leq l$ . We say that symbol  $a$  occurs at position  $i$  of string  $w$  if  $w = w_1aw_2$  and  $|w_1| = i-1$ .



each  $i$ ,  $1 < i \leq n$ , the length of the longest substring of  $\text{result}(T_i^{1i})$  which contains distinguished positions only is less than or equal to  $q$ .

(b)  $z$  can be represented as:

$$z = u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n, \text{ where}$$

(i)  $(v_1 w_1 x_1, \dots, v_n w_n x_n)$ , is the  $n$ -result of the subtree of  $T$  rooted at  $M_1$ , and

(ii)  $(w_1, \dots, w_n)$ , is the  $n$ -result of the subtree rooted at  $M_2$ , such that

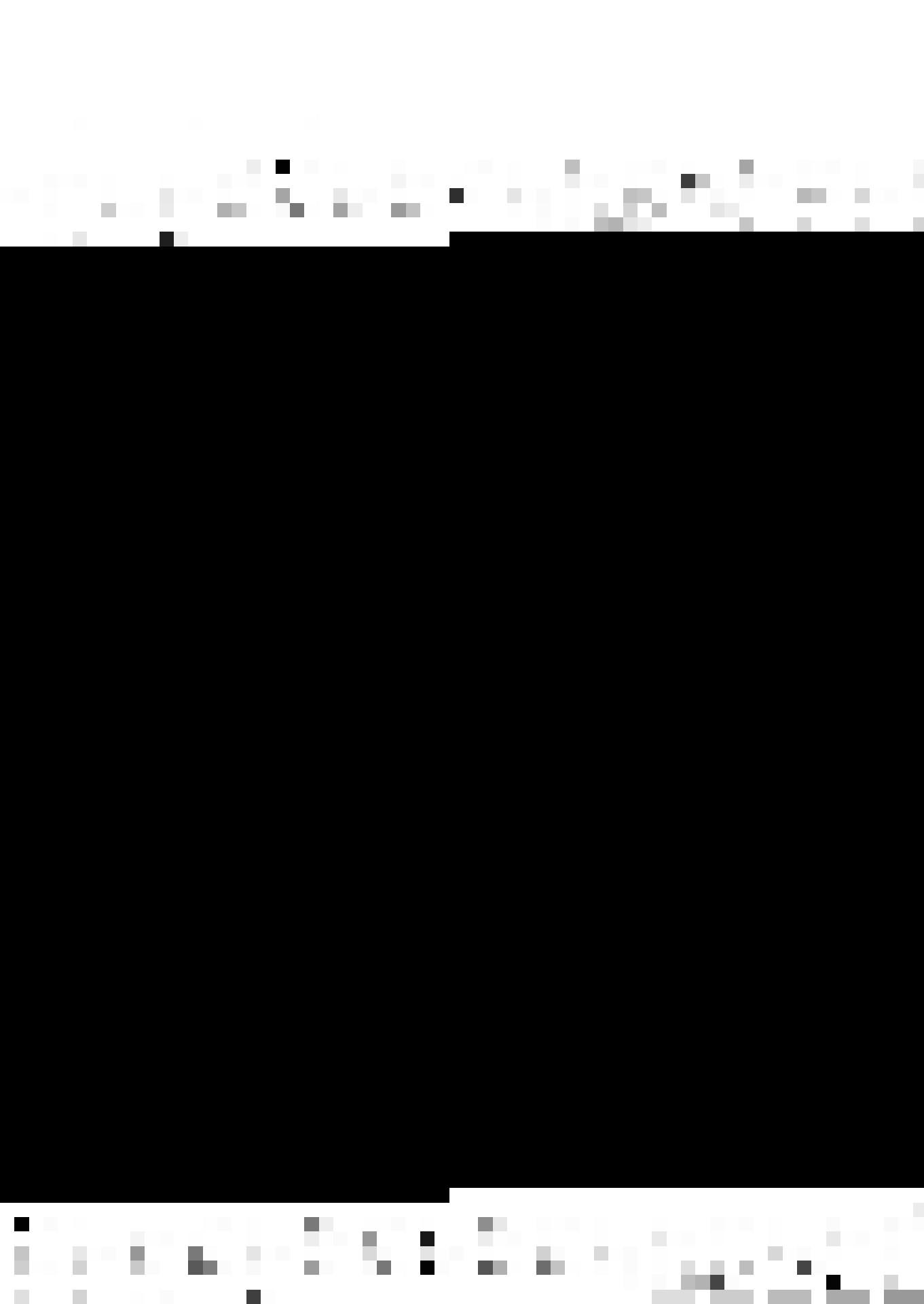
$$u_1 v_1^i w_1^i x_1^i y_1 \dots u_n v_n^i w_n^i x_n^i y_n, \text{ is also in } L \text{ for all } i \geq 0.$$

Proof:

We will only prove part (a) of the theorem. Once part (a) is proved, the proof of part (b) is on the same lines as the proof of the uvwxy theorem for context-free languages(18).

Let  $G = (V_N, V_T, P, (S)^n)$ . Let  $m$  be the number of nonterminals in  $V_N$ .

The first thing that we must now do is to say how the values of  $p$  and  $q$  are to be determined. The general idea is to choose  $p$  such that there is an  $n$ -path in  $T$  having two  $n$ -nodes on it with the same labels. Actually, there is more in it, than above, because of the various conditions specified in a(ii), which are to be satisfied.



We specify  $p$  and  $q$  as follows:  $p = k^{m-1}$  and  $q = k^m$ , if  $k \geq 1$ .  $p = q = 1$ , if  $k = 0$  (note that  $|z_1| > n.p$  is not possible in case  $k = 0$ ).

Let  $k \geq 1$ . Now, we show, through a sequence of steps that with the above choice of  $p$  and  $q$ , the conditions specified in (a) are indeed satisfied:

1. It is easy to see that for atleast one  $i$ ,  $1 \leq i \leq n$ , there exists  $z_2$ , a substring of  $\text{result}(T_i)$ , such that  $z_2$  contains distinguished positions only and  $|z_2| > p$ . Let this be the case for  $i$  in  $\{i_1, \dots, i_t\}$ ,  $t \geq 1$ .

Note that if  $t > 1$ , then  $i_1, \dots, i_t$  need not refer to a consecutive set of trees comprising  $T$ .

2. Let  $i$  be in  $\{i_1, \dots, i_t\}$ . We observe that the maximum number of direct descendants of a node in any  $T_i$  is  $k$ . Since  $|z_2| > p = k^{m-1}$ , it follows, using Lemma 2.1.2, that there is a path in  $T_i$  of length greater than  $m$ ; there must surely be two nodes with the same label on such a path, for  $m$  is the number of nonterminals in  $V_N$ .

3. Now, for  $i$  in  $\{i_1, \dots, i_t\}$ , we show that there is atleast one choice of nodes  $N_{1i}$  and  $N_{2i}$  in  $T_i$  such that:

their labels are the same;

$N_{2i}$  is a descendant of  $N_{1i}$ ; and

either  $l\text{-result}(T_i^{N_{1i}-N_{2i}})$  or  $r\text{-result}(T_i^{N_{1i}-N_{2i}})$

contains a distinguished position.

Call such a choice as  $\gamma$ .



Suppose for the time being that there does not exist any  $\gamma$  for  $T_i$ . We have already shown that there exists a path in  $T_i$  and two nodes, say,  $N_1$  and  $N_2$  on the path, having the same label. Let us reduce  $T_i$  by replacing the subtree rooted at  $N_1$  by that rooted at  $N_2$  (assuming that  $N_2$  is a descendant of  $N_1$ ). Call this as a 'reduction' of  $T_i$ . Say that  $T_{i1}$  is the result of a reduction of  $T_i$  and  $T_{is}$ ,  $s \geq 2$ , is the result of  $s$  reductions of  $T_i$ , if there exist  $T_{i1}, \dots, T_{is}$ , such that  $T_{ir}$ ,  $2 \leq r \leq s$ , is the result of a reduction of  $T_{i(r-1)}$ .

We observe that there does not exist  $\gamma$  for any tree obtained from  $T_i$  by  $s$ ,  $s \geq 1$ , number of reductions because otherwise  $\gamma$  would exist for  $T_i$  also, which contradicts our supposition. Now, let  $T'_i$  be the result of  $s'$  reductions of  $T_i$  such that there is no path in  $T'_i$  having two nodes on it with the same label. Evidently,  $z_2$  (containing distinguished positions only) is a substring of  $\text{result}(T'_i)$ , and therefore  $|\text{result}(T'_i)| > p = k^{m-1}$ . Now, using Lemma 2.1.2, it follows that there is a path in  $T'_i$  having two nodes on it with the same label — a contradiction. This proves what we wanted to show in this step.

4. Let  $X$  be the set of all  $i$ ,  $1 \leq i \leq n$ , such that for each  $i$  in  $X$ , there is a path  $P_i$  in  $T_i$  and two nodes  $M_1^i$  and  $M_2^i$  on the path having the following properties:



- (i)  $P_i$  is an  $(M_1^i, M_2^i)$ -related path in  $T_i$ ;
- (ii) Either l-result( $T_i^{M_1^i-M_2^i}$ ) or r-result( $T_i^{M_1^i-M_2^i}$ ) contains a distinguished position;
- (iii)  $M_1^i$  is the farthest such node from the root of  $T_i$  (the choice need not be unique).

Note that  $X$  is a subset of  $\{i_1, \dots, i_t\}$ , and is nonempty in view of 3 above.

Let  $j$  in  $X$  be such that among the paths  $R_i, \dots, M_1^i$ , where  $i$  is in  $X$  and  $R_i$  is the root of  $T_i$ , the length of the path  $R_j, \dots, M_1^j$  is maximum ( $j$  need not be unique). (For the significance of  $j$  and the  $n$ -nodes  $M_1$  and  $M_2$  which we will presently define, see the statement of the theorem). Let

$M_1 = < M_{11}, \dots, M_{1n} >$ , and  $M_2 = < M_{21}, \dots, M_{2n} >$ , where  $M_{1j}$  is the same node as  $M_1^j$  and  $M_{2j}$  is the same node as  $M_2^j$ .

5. We now show that for each  $i$ ,  $1 \leq i \leq n$ , the length of the longest substring of result( $T_i^{M_1^i}$ ) which contains distinguished positions only, is less than or equal to  $q$ .

Make 'reductions' (defined in step 3 of the proof) in  $T_i^{M_1^i}$ , which do not involve  $M_{1i}$  in the following sense; at no stage should the subtree rooted at  $M_{1i}$  be replaced by some other subtree, till a tree  $T_i''$  is obtained in which in any path, no more than two nodes (in which case one of the nodes should be  $M_{1i}$ ) have got the same label. From the way the  $n$ -node  $M_1$  has been chosen (see step 4), it is clear that any substring of



$\text{result}(T_i^{M_1})$  which contains distinguished positions only, will be a substring of  $\text{result}(T_i'')$ . For otherwise<sup>1</sup> the choice of  $M_1$  is not in accordance with step 4. Now observing that there can be no path in  $T_i''$  of length greater than  $m+1$ , the result, which we wanted to prove in this step, follows using Lemma 2.1.2.

This completes the proof of part (a) of the theorem. The proof of part (b) is now on the same lines as that of uvwxy theorem for context-free languages and is omitted.

Q.E.D.

Cor. 2.1.1.1:

Let  $L$  be an  $(n, \sigma, k)$ CL (or  $n$ CL). There exist constants  $p$  and  $q$ , depending only on  $L$ , such that if there is a sentence  $z$  in  $L$  with  $|z| > n.p$ , then  $z$  may be written as:

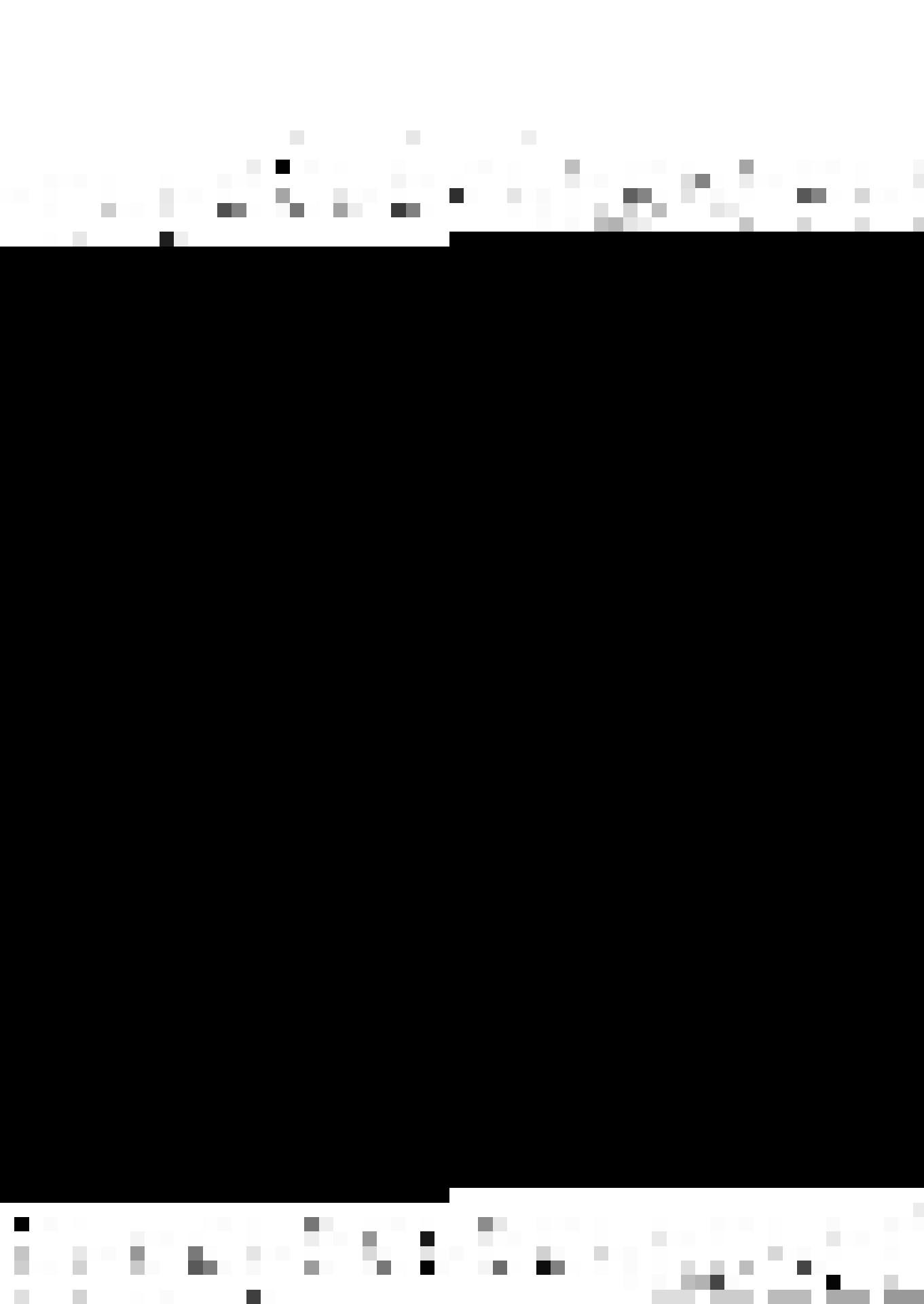
$$z = u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n, \text{ where } v_1 x_1 \dots v_n x_n \neq \epsilon,$$

and  $|v_i w_i x_i| \leq q$ ,  $1 \leq i \leq n$ , such that

$$u_1 v_1^i w_1^i x_1^i y_1 \dots u_n v_n^i w_n^i x_n^i y_n, \text{ is also in } L \text{ for all } i \geq 0.$$

\*\*\*

1. Suppose that there is a substring of  $\text{result}(T_i^{M_1})$  which contains distinguished positions only but is not a substring of  $\text{result}(T_i'')$ . This implies that there is an  $(N_1, N_2)$ -related path in  $T_i^{M_1}$  such that either  $l\text{-result}(T_i^{N_1-N_2})$  or  $r\text{-result}(T_i^{N_1-N_2})$  contains a distinguished position;  $N_1$  being different from and a descendant of  $M_1$ . This implies that the choice of  $M_1$  is not in accordance with step 4.



In view of the fact that every context-free language is an  $(1, \emptyset, k)$ CL for some  $k$ , we have:

Cor. 2.1.1.2:

Let  $L$  be a context-free language. There exist constants  $p$  and  $q$ , depending only on  $L$ , such that if there is a sentence  $z$  in  $L$ ,  $z_1$  is a substring of  $z$ ,  $|z_1| > p$ , and the positions occupied by the symbols comprising  $z_1$  are designated as 'distinguished', then  $z$  may be written as:

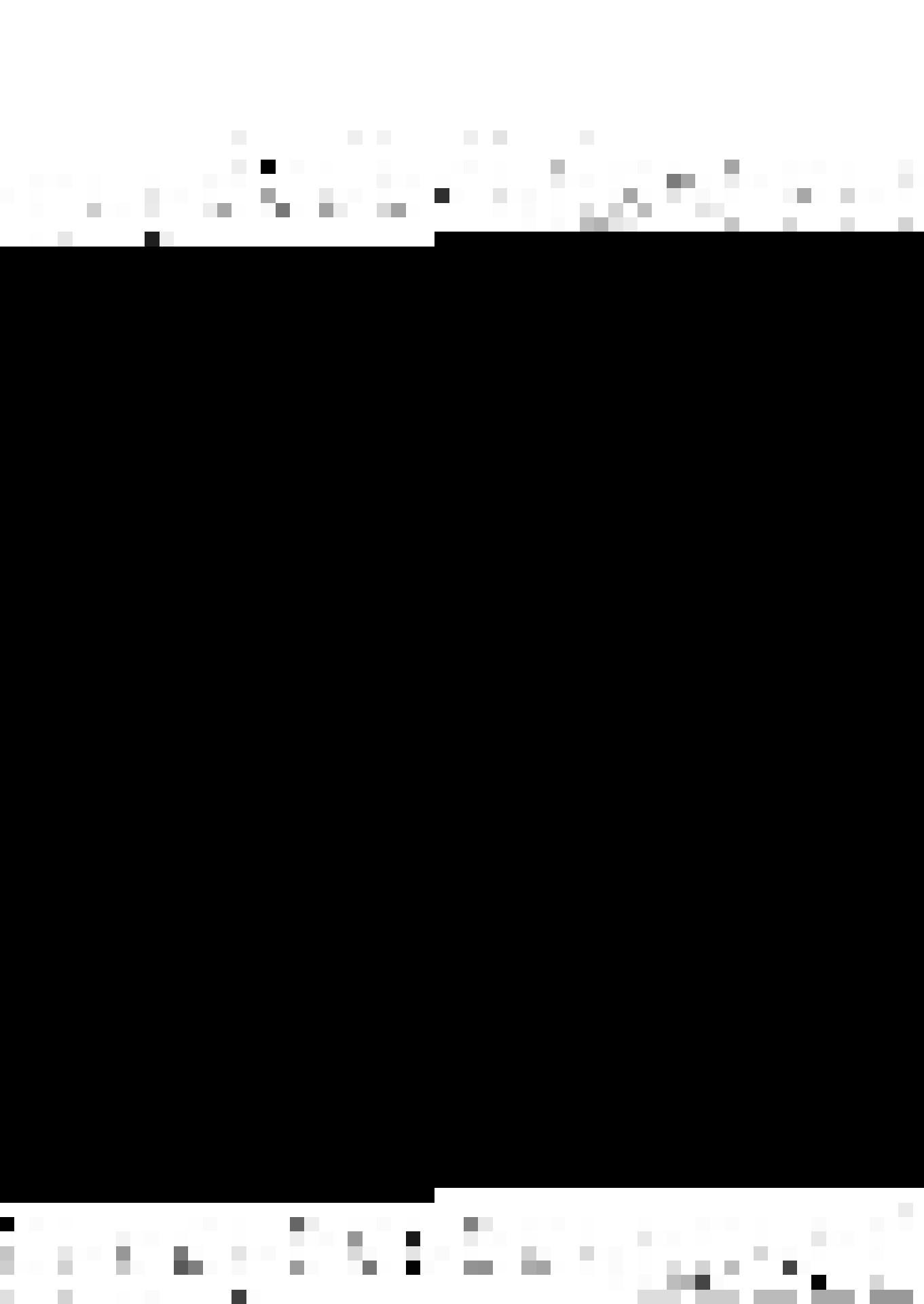
$z = uvwxy$ , where  $v$  or  $x$  contains a distinguished position, and the length of the longest substring of  $vwx$  which contains distinguished positions only is less than or equal to  $q$ , such that

$uv^{i_1}wx^{i_2}y$  is in  $L$  for all  $i \geq 0$ .

\*\*\*

Observe that the above result for context-free languages is more powerful than the  $uvwxy$  theorem in that it allows decomposition relative to any given substring  $z_1$ . We can actually look at Theorem 2.1.1 as a generalization of this result to  $(n, \sigma, k)$ -coupled languages.

Now, we introduce some more concepts related to  $n$ -derivation trees and labeled trees and give some results concerning these concepts. We shall also give an example to illustrate these concepts.



Two paths (respectively, n-paths)  $P_1$  and  $P_2$  are said to be  $((M_1, N_1), (M_2, N_2))$ -mutually distinct in a rooted labeled tree (respectively, an n-derivation tree)  $T$  if:

- (a)  $P_1$  is an  $(M_1, N_1)$ -related path (respectively, n-path) and is to the left of  $P_2$  which is an  $(M_2, N_2)$ -related path (respectively, n-path).
- (b) Neither of  $M_1$  and  $N_1$  is on  $P_2$ . Neither of  $M_2$  and  $N_2$  is on  $P_1$ .

A sequence of paths (respectively, n-paths)  $P_1, \dots, P_m$  are said to be  $((M_1, N_1), \dots, (M_m, N_m))$ -mutually distinct in  $T$  if for every pair  $(i, j)$ ,  $1 \leq i < j \leq m$ ,  $P_i$  and  $P_j$  are  $((M_i, N_i), (M_j, N_j))$ -mutually distinct paths (respectively, n-paths) in  $T$ .

\*\*\*

The validity of the following result is easy to see.

#### Lemma 2.1.3:

Let  $T$  be a rooted labeled tree having an  $(M_i, N_i)$ -related path for every  $i$ ,  $1 \leq i \leq m$ .

There are  $((M_1, N_1), \dots, (M_m, N_m))$ -mutually distinct paths in  $T$  iff  $\text{result}(T)$  can be written as:

$x_0 Y_1 Y_2 \dots Y_m$ , where  $Y_i = \text{result}(T^{\overset{M_i}{\overbrace{i}}}) x_i$ ,  $1 \leq i \leq m$ , and  $x_0, \dots, x_m$  are certain strings (possibly empty).

\*\*\*



The following result follows from the definition of n-derivation trees.

Lemma 2.1.4:

Let  $1 \leq i \leq m$ . Let  $M_i = \langle M_{i1}, \dots, M_{in} \rangle$  and  $N_i = \langle N_{i1}, \dots, N_{in} \rangle$  be n-nodes in an n-derivation tree  $T = \langle T_1, \dots, T_n \rangle$ . If for some  $j$ ,  $1 \leq j \leq n$ , there are  $((M_{1j}, N_{1j}), \dots, (M_{mj}, N_{mj}))$ -mutually distinct paths in  $T_j$ , then there are  $((M_{i_1}, N_{i_1}), \dots, (M_{i_m}, N_{i_m}))$ -mutually distinct n-paths in  $T$ , where the sequence  $i_1, \dots, i_m$  is some permutation over  $1, \dots, m$ .

\*\*\*

We now have a definition which indicates how n-paths in an n-derivation tree are 'distributed' among the trees comprising the n-derivation tree:

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an n-derivation tree,  $n \geq 2$ , The  $(i_1, \dots, i_s)$ -distribution,  $1 \leq i_1 < \dots < i_s \leq n$ ,  $s \geq 2$ , of the  $m (\geq 1)$  n-paths:  $\langle p_{11}, \dots, p_{n1} \rangle, \dots, \langle p_{1m}, \dots, p_{mm} \rangle$  in  $T$ ,  $p_{i_11}, \dots, p_{i_1m}$  being paths in  $T_{i_1}$  in order from the left, is said to be  $(\kappa_{i_2}, \dots, \kappa_{i_s})$ , where  $\kappa_{i_2}, \dots, \kappa_{i_s}$  are certain permutations on  $m$  objects, if for  $i \in \{i_2, \dots, i_s\}$ ,  $p_{ij}$ ,  $1 \leq j \leq m$ , is the  $\kappa_i(j)$ -th path, in order from the left, among the paths  $p_{i_11}, \dots, p_{i_1m}$  in  $T_i$ .

\*\*\*



The following result follows from the definition of the  $\mathcal{U}$ -simplicity of  $n$ -coupled grammars and the definition of  $n$ -derivation trees.

Lemma 2.1.5:

Let  $G$  be an  $(n, \sigma, k)$ CG, where  $\sigma = \xi_1, \dots, \xi_{n-1}$ . If in an  $n$ -derivation tree (in  $G$ ) of a sentence in  $L(G)$ , the  $(j_1, \dots, j_s)$ -distribution of some  $n$ -paths is  $(\pi_{j_1}, \dots, \pi_{j_s})$ , then  $\pi_m$  is an identity permutation, where  $m$  is in  $\{j_2, \dots, j_s\} \cap \xi_{j_1}$ .

\*\*\*

Now we give an example which will illustrate the concepts introduced above.

Example 2.1.1:

Let  $G$  be the  $(3, \{3\}, \emptyset, 2)$ CG of Example 1.1.4. Figure 2.1.1 is a representation of a 3-derivation tree in  $G$ . For the sake of reference, the nodes have been numbered. The labels and permutations associated with the nodes (if any) are shown adjacent to the nodes.

Let  $p_{11} = 1, 2, 5, 11$ ,  $p_{12} = 1, 3, 8$ ,  $p_{21} = 17, 18, 21$ ,  $p_{22} = 17, 19, 24, 29$ ,  $p_{31} = 33, 34, 37, 43$  and  $p_{32} = 33, 35, 40$ , be some of the paths in the three trees.

$P_1 = \langle p_{11}, p_{22}, p_{31} \rangle$  and  $P_2 = \langle p_{12}, p_{21}, p_{32} \rangle$  are two 3-paths and,







$M_1 = \langle 2, 19, 34 \rangle$ ,  $M_2 = \langle 5, 24, 37 \rangle$ ,  
 $M_3 = \langle 11, 29, 43 \rangle$ ,  $M_4 = \langle 3, 18, 35 \rangle$ , and  
 $M_5 = \langle 8, 21, 40 \rangle$ , are some of the 3-nodes, in  
the 3-derivation tree.

We notice that  $P_1$  and  $P_2$  are  $((M_1, M_2), (M_4, M_5))$ -  
mutually distinct 3-paths. They are also  $((M_1, M_3), (M_4, M_5))$ -  
mutually distinct as well as  $((M_2, M_3), (M_4, M_5))$ -mutually dis-  
tinct, 3-paths.

Again, we notice that the  $(1, 2, 3)$ -distribution of  $P_1$  and  
 $P_2$  is  $([2, 1], [1, 2])$ , their  $(2, 3)$ -distribution is  $([2, 1])$  and  
their  $(1, 3)$ -distribution is  $([1, 2])$ .

\*\*\*

Now, we shall define the concepts of 'language simplicity predicate' and 'translation simplicity predicate' which play an important role in proving the structure results in this chapter, particularly those results which involve the  $\mu$ -simplicity of coupled grammars.

The language simplicity predicate  $\mathcal{P}^L$  is a mapping from the cross product of the  $\mu$ -simplicity of coupled grammars into the set  $\{0, 1\}$ . The basic results that we prove using the concept of language simplicity predicate are:

1. Every language in  $\mathcal{C}_{(n, \sigma, k)}^L$  is also in  $\mathcal{C}_{(n', \sigma', k')}^L$ , where  $n' \geq n$ ,  $k' \geq k$  and  $\mathcal{P}^L(\sigma', \sigma) = 1$  (Theorem 2.2.1); and



2. There is a language in  $\mathcal{C}_{(n, \sigma, k)}^L$ ,  $n \geq 2, k \geq 2$ , which is not in  $\mathcal{C}_{(n', \sigma')}^L$ , where  $n' > n$  and  $\rho^L(\sigma', \sigma) = 0$  (Lemma 2.3.4).

We define the language simplicity predicate, formally, as follows:

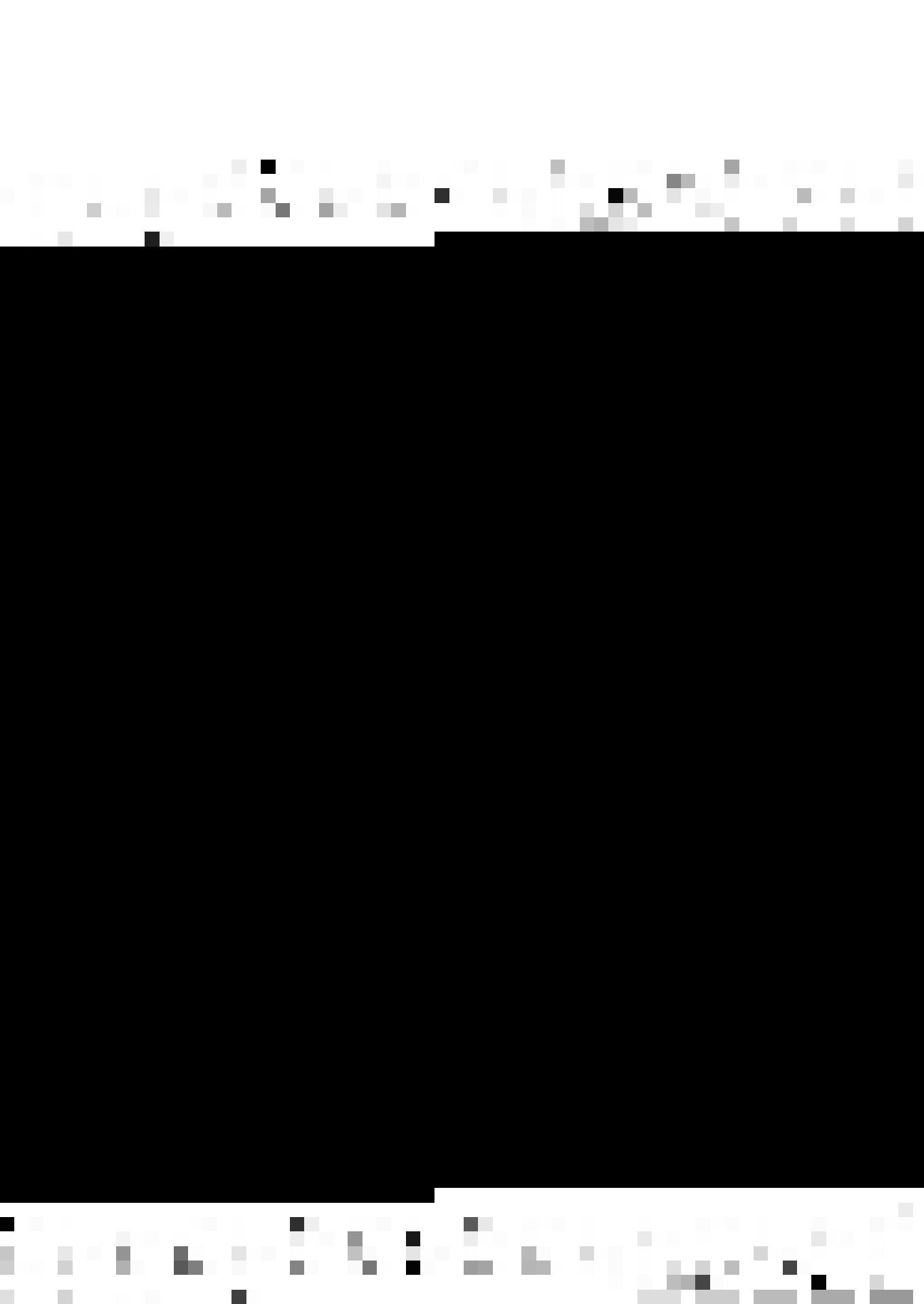
Let  $\sigma' = \xi'_1, \dots, \xi'_{n'-1}$ , be the  $\mathcal{U}$ -simplicity of some  $n'CG G'$  and let  $\sigma = \xi_1, \dots, \xi_{n-1}$ , be the  $\mathcal{U}$ -simplicity of some  $nCG G$ . The language simplicity predicate  $\rho^L$  is defined as follows:

$\rho^L(\sigma', \sigma) = 1$ , iff there exist integers  $i_1, \dots, i_n$ , where  $1 \leq i_1 < \dots < i_n \leq n'$ , such that for every  $j$ ,  $1 \leq j \leq n-1$ , if  $s$  in  $\{j+1, \dots, n\}$ , is not in  $\xi_j$ , then  $i_s$  is not in  $\xi'_{i_j}$ , and  $\rho^L(\sigma', \sigma) = 0$ , otherwise.

\*\*\*

Note that  $\rho^L(\sigma', \sigma) = 0$ , iff for every possible choice  $\psi$  of the sequence  $i_1, \dots, i_n$ , such that  $1 \leq i_1 < \dots < i_n \leq n'$ , there is atleast one  $j$ ,  $1 \leq j \leq n-1$ , and atleast one  $s$  in  $\{j+1, \dots, n\}$ , the values of  $j$  and  $s$  possibly depending on  $\psi$ , such that  $s$  is not in  $\xi_j$ , while  $i_s$  is in  $\xi'_{i_j}$ .

Using the definition of the translation simplicity predicate, certain structure results follow from corresponding results for coupled languages. This point will become clear after we define the concept of translation simplicity predicate.



Let  $\sigma' = \xi'_1, \dots, \xi'_{m'+n'-1}$ , be the  $\mathcal{U}$ -simplicity of some  $(m'+n')\text{CG } G'$  and let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ , be the  $\mathcal{U}$ -simplicity of some  $(m+n)\text{CG } G$ , where  $m, n, m', n' \geq 1$ . We define the translation simplicity predicate  $\rho^T$  as follows:

$\rho^T(\sigma', \sigma, m', m) = 1$ , iff there exist integers

$i_1, \dots, i_{m+n}$ , where  $1 \leq i_1 < \dots < i_{m+n} \leq m'+n'$ ,  $i_m \leq m'$ , and  $i_{m+1} \geq m'+1$ , such that for every  $j$ ,  $1 \leq j \leq m+n-1$ , if  $s$  in  $\{j+1, \dots, m+n\}$ , is not in  $\xi_j$ , then  $i_s$  is not in  $\xi'_{i_j}$ , and  $\rho^T(\sigma', \sigma, m', m) = 0$ , otherwise.

\*\*\*

Note that  $\rho^T(\sigma', \sigma, m', m) = 1$  implies that  $\rho^L(\sigma', \sigma) = 1$ , and  $\rho^L(\sigma', \sigma) = 0$  implies that  $\rho^T(\sigma', \sigma, m', m) = 0$ . It is this fact which makes certain structure results for coupled translations follow from corresponding results for coupled languages.

From the definition of the language simplicity predicate, the following result follows:

#### Lemma 2.1.6:

Let  $\sigma' = \xi'_1, \dots, \xi'_{n'-1}$  be the  $\mathcal{U}$ -simplicity of some  $n'\text{CG } G'$  and let  $\sigma = \xi_1, \dots, \xi_{n-1}$  be the  $\mathcal{U}$ -simplicity of some  $n\text{CG } G$ . We have the following:

- (a) If  $n' < n$ , then  $\rho^L(\sigma', \sigma) = 0$ .
- (b) Let  $n' = n$ . Then,



(i)  $\rho^L(\sigma', \sigma) = 1$ , iff for every  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \subseteq \xi_j$ .

(ii)  $\rho^L(\sigma', \sigma) = 1$ , and  $\rho^L(\sigma, \sigma') = 0$ , iff for every  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \subseteq \xi_j$ , and for at least one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \not\subseteq \xi_j$ .

(iii)  $\rho^L(\sigma', \sigma) = 0$  and  $\rho^L(\sigma, \sigma') = 0$ , iff either of the following two statements is true:

1. For at least one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \not\subseteq \xi'_j$ , and for at least one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \not\subseteq \xi_j$ .

2. For at least one  $j$ ,  $\xi'_j$  and  $\xi'_j$  are incomparable.

(c) If  $n = 1$ , then  $\rho^L(\sigma', \sigma) = 1$ .

(d) If  $G'$  is an  $n'$ SCG and  $n' \geq n$ , then  $\rho^L(\sigma', \sigma) = 1$ , iff  $G$  is an  $n$ SCG.

\*\*\*

The following example illustrates the concept of the language simplicity predicate  $\rho^L$ .

#### Example 2.1.2:

Let the  $\mathcal{U}$ -simplicity of some 4CG be  $\sigma' = \xi'_1, \xi'_2, \xi'_3$ , where  $\xi'_1 = \{4\}$ ,  $\xi'_2 = \{3\}$ , and  $\xi'_3 = \emptyset$ . Let the  $\mathcal{U}$ -simplicity of some 3CG be  $\sigma = \xi_1, \xi_2$ , where  $\xi_1 = \{2\}$  and  $\xi_2 = \emptyset$ .

We observe that  $\rho^L(\sigma', \sigma) = 1$ , because there exist integers  $i_1 = 2$ ,  $i_2 = 3$ , and  $i_3 = 4$ , such that for every  $j$ ,  $1 \leq j \leq 2$ , if  $s$  in  $\{j+1, \dots, 3\}$  is not in  $\xi_j$ , then



$i_s$  is not in  $\xi'_{i_j}$  as follows: For  $j = 1$ , 3 is not in  $\xi'_1$ ; 4 is not in  $\xi'_2$ . For  $j = 1$ , 3 is not in  $\xi'_2$ ; 4 is not in  $\xi'_3$ .

On the other hand in the above example, let the  $\mu$ -simplicity of the 3CG be  $\sigma'' = \xi''_1, \xi''_2$ , where  $\xi''_1 = \xi''_2 = \emptyset$ .

We observe that  $P^L(\sigma', \sigma'') = 0$ , because for each choice of  $i_1, i_2, i_3$  viz. (i)  $i_1 = 1, i_2 = 2, i_3 = 3$ , (ii)  $i_1 = 1, i_2 = 3, i_3 = 4$ , and (iii)  $i_1 = 2, i_2 = 3, i_3 = 4$ , there is atleast one  $j$ ,  $1 \leq j \leq n-1$ , and atleast one  $s$  in  $\{j+1, \dots, n\}$ , such that  $s$  is not in  $\xi''_j$  while  $i_s$  is in  $\xi'_{i_j}$ , as follows:

(i)  $i_1 = 1, i_2 = 2, i_3 = 3$ .

Here,  $j = 2$ , and  $s = 3$ . While 3 is not in  $\xi''_2$ ,  $i_3 = 3$  is in  $\xi'_{i_2} = \xi'_2$ .

(ii)  $i_1 = 1, i_2 = 3, i_3 = 4$ .

Here,  $j = 1$ , and  $s = 3$ . While 3 is not in  $\xi''_1$ ,  $i_3 = 4$  is in  $\xi'_{i_1} = \xi'_1$ .

(iii)  $i_1 = 2, i_2 = 3, i_3 = 4$ .

Here,  $j = 1$ , and  $s = 2$ . While 2 is not in  $\xi''_2$ ,  $i_2 = 3$  is in  $\xi'_{i_1} = \xi'_1$ .

## 2.2 STRUCTURE RESULTS - I:

In this section, we shall concern ourselves with the structure induced on coupled languages and coupled translations by the degree of the coupled grammars generating them. We shall see that the degree of coupled grammars defines an infinite hierarchy in the class of coupled languages and coupled translations.

### Hierarchies Based on Degree:

Before we come to the main results of this section, we shall prove some basic results (Lemma 2.2.1, Lemma 2.2.2 and Theorem 2.2.1):

#### Lemma 2.2.1:

There is a language L which is in  $\mathcal{C}_{(n+1, \sigma, k)}^L$ , but is not in  $\mathcal{C}_n^L$ ,  $k \geq 1$ .

#### Proof:

Let  $L = \{a_1^i \dots a_{2n+1}^i \mid i \geq 1\}$ . It can easily be shown that it is an  $(n+1, \sigma, k)CL$ . (See Appendix A).

We shall now show that L is not an nCL. First, let us make a few observations about the nature of the language L. We observe that in every sentence in L,

- (a) the number of occurrences of each of the  $2n+1$  distinct symbols, viz.  $a_1, \dots, a_{2n+1}$ , is the same, and

(b) for each distinct symbol, all the occurrences of that symbol are adjacent to each other.

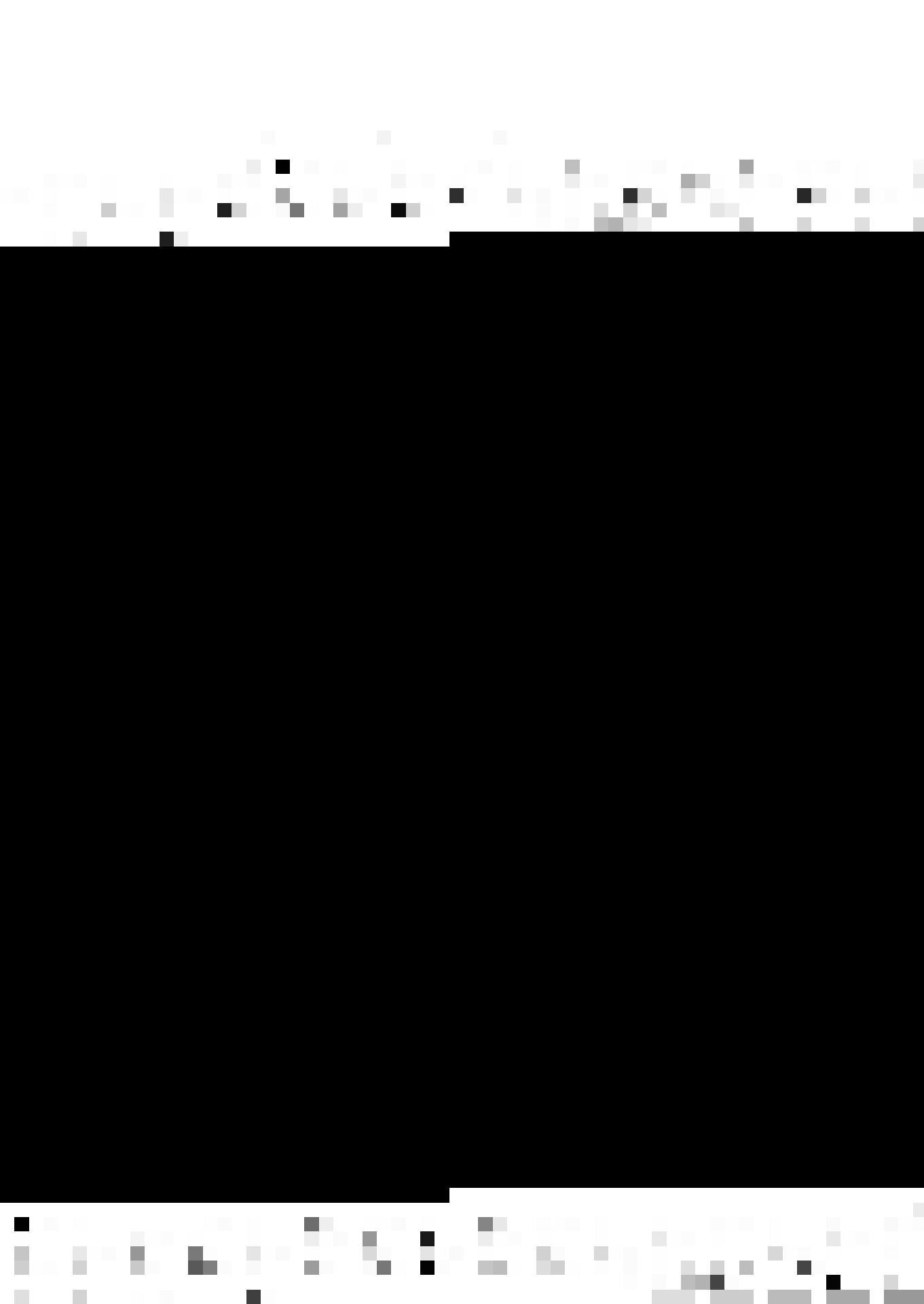
Now, let us suppose for the time being that  $L$  is an nCL. Let it be an  $(n, \sigma', k')$  CL, for some  $\sigma'$  and  $k'$ . Consider the sentence  $z = a_1^p \dots a_{2n+1}^p$  in  $L$ , where  $p$  is the constant referred to in Cor. 2.1.1.1. We observe that  $|z| > n.p$ , and therefore we have, from the above mentioned corollary:  $z$  can be represented as,

$z = u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n$ , where  $v_1 x_1 \dots v_n x_n \neq \epsilon$ , such that for each integer  $i \geq 0$ ,  $z(i) = u_1 v_1^i w_1^i x_1^i y_1 \dots u_n v_n^i w_n^i x_n^i y_n$ , is also in  $L$ . In view of this and (b) above, it follows that none of  $v_1, x_1, \dots, v_n, x_n$ , can have occurrences of more than one distinct symbol. Now, since there are occurrences of  $2n+1$  distinct symbols in  $z$ , there will be atleast one symbol which does not occur in any of  $v_1, x_1, \dots, v_n, x_n$ . This easily leads to a contradiction in view of (a) above and the fact that  $z(i)$  is in  $L$  for all  $i \geq 0$ . Therefore,  $L$  is not an nCL.

Hence the result.

Q.E.D.

In view of the fact that the language  $L$  in the proof of the above lemma is an  $(n+1, k)$  SCL,  $k \geq 1$ , as well, we have:



Lemma 2.2.2:

There is a language L which is in  $\mathcal{C}_{(n+1, k)}^L$ ,  $k \geq 1$ , but is not in  $\mathcal{C}_n^L$ .

Theorem 2.2.1:

$\mathcal{C}_{(n', \sigma', k')}^L \supset \mathcal{C}_{(n, \sigma, k)}^L$ , where  $n' > n$ ,  $k' \geq k$  and  $\mathcal{P}^L(\sigma', \sigma) = 1$ .

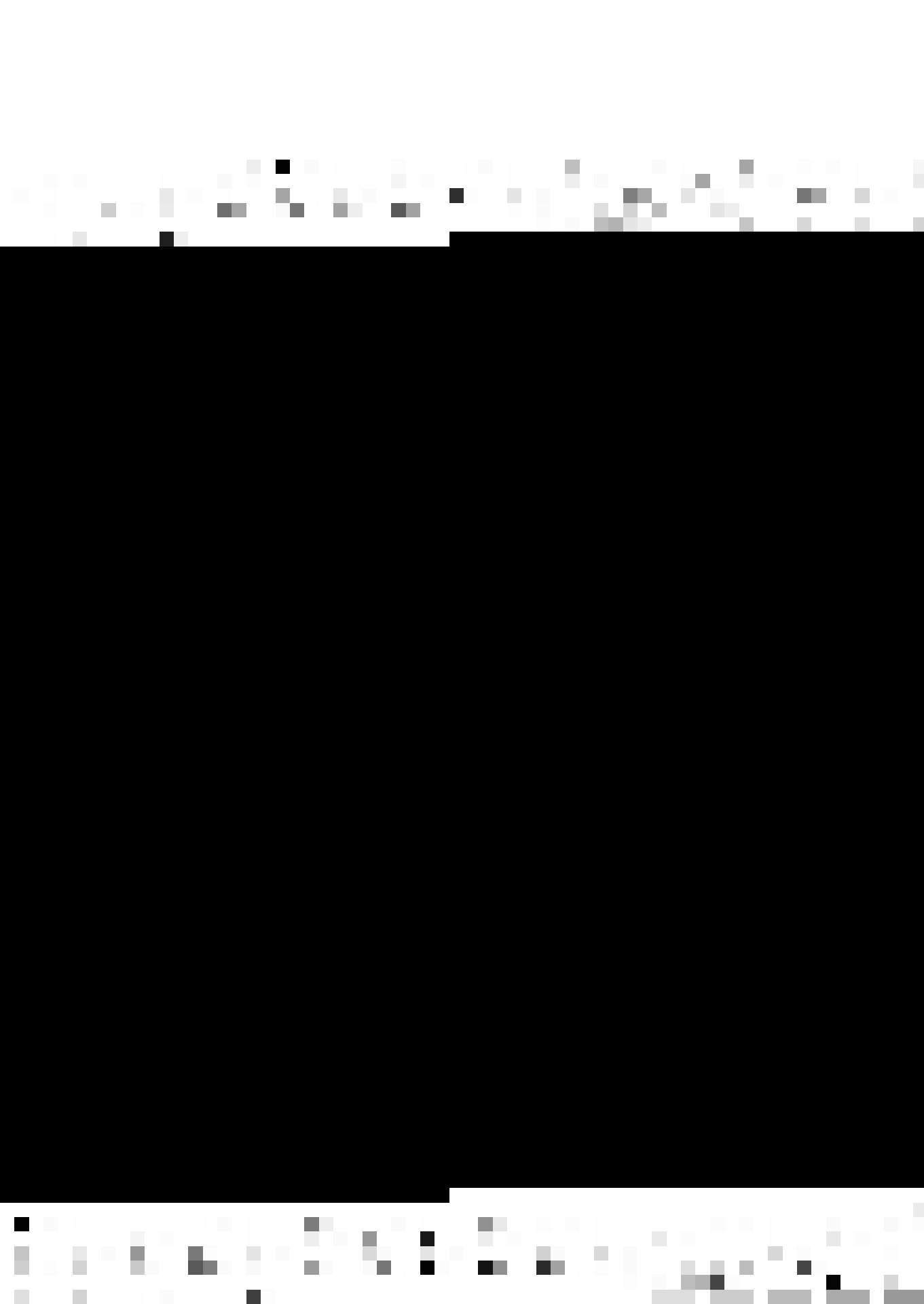
Proof:

Let  $k' \geq 1$ . Let  $\sigma' = \xi'_1, \dots, \xi'_{n'-1}$  and  $\sigma = \xi_1, \dots, \xi_{n-1}$ . Since  $\mathcal{P}^L(\sigma', \sigma) = 1$ , it follows that there exist integers  $i_1, \dots, i_n$ , where  $1 \leq i_1 < \dots < i_n \leq n'$ , such that for every  $j$ ,  $1 \leq j \leq n-1$ , if  $s$  in  $\{j+1, \dots, n\}$ , is not in  $\xi_j$ , then  $s$  is not in  $\xi'_{i_j}$ .

Consider any language L in  $\mathcal{C}_{(n, \sigma, k)}^L$ . Let it be generated by an  $(n, \sigma, k)$ CG  $G = (V_N, V_T, P, (S)^n)$ . To prove the result, it suffices to prove that L can be generated by some  $(n', \sigma', k')$ CG. We shall construct  $G'$  using the algorithm given below. This algorithm has got similarity with the algorithm given in the proof of Theorem 1.1.1.

The idea behind the algorithm is as follows: Let there be one production in  $G'$  corresponding to each production in  $G$ ,

- 
1. In case  $k = 0$ , it is easy to prove the result, all that has to be done is to add a suitable dummy production.



which is such that it does not make it impossible<sup>1</sup> for  $G'$  to have the order  $k'$  and the  $\mu$ -simplicity  $\sigma'$ , and such that because of these productions  $G'$  generates the language  $L$ . Let the fact that the  $\mu$ -simplicity and the order of  $G'$ , have to be  $\sigma'$  and  $k'$ , respectively, be taken care of, by having some production(s) in  $G'$  which are not involved in the generation of any sentence by  $G'$ .

#### Algorithm:

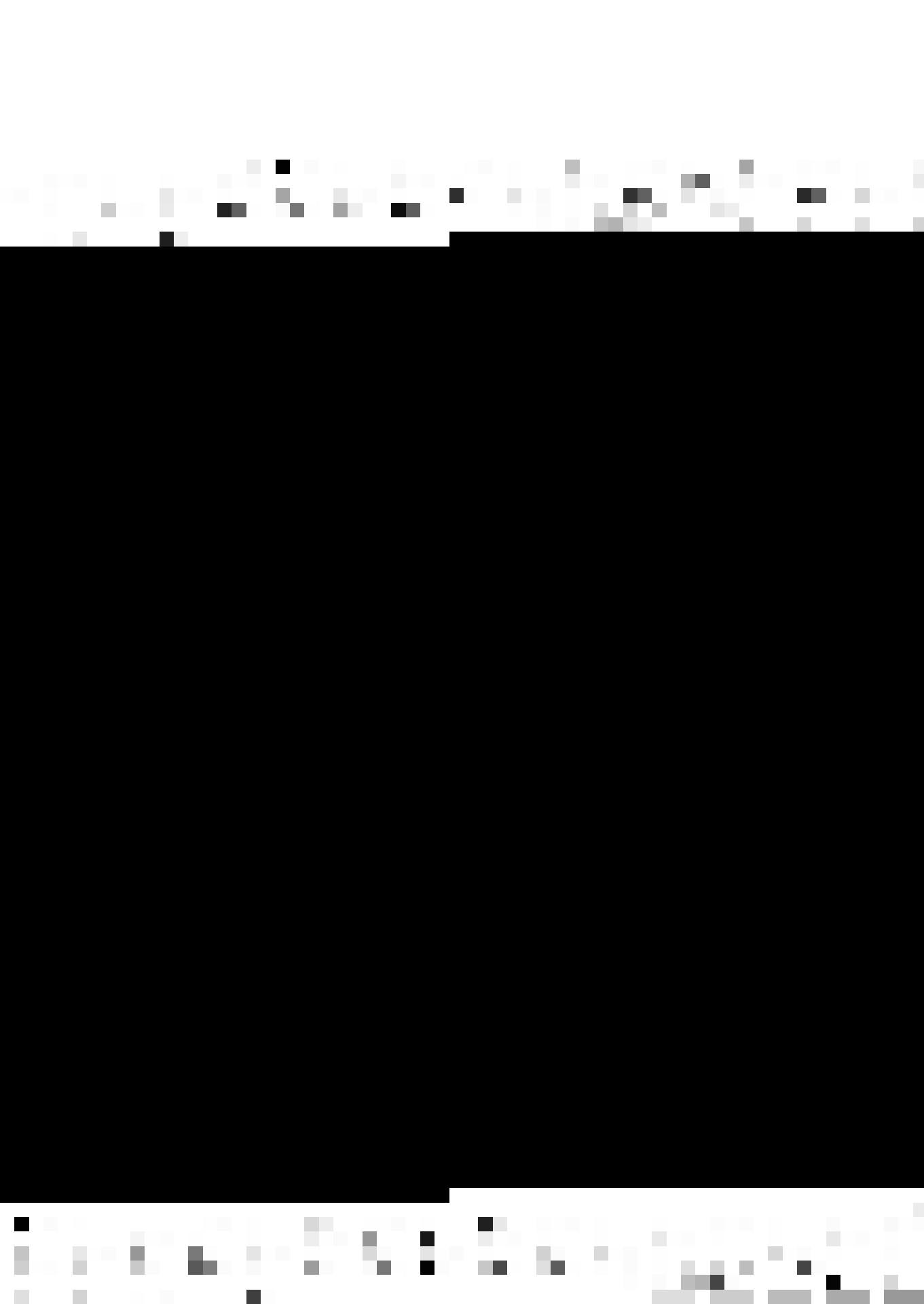
We shall intersperse the algorithm with comments, which we will keep within parentheses. We shall use some of these comments later in proving that  $G'$  is an  $(n', \sigma', k')$ CG and that  $L(G') = L$ .

If  $k' \geq 2$ , let  $G'' = (V_N'', \emptyset, P'', (S_1)^{n'})$ , be the sample  $(n', \sigma', k')$ CG (see the proof of Theorem 1.1.1) such that  $V_N'' \cap V_R = \emptyset$ .

Now,  $G' = (V_R', V_T, P', (S)^{n'})$ , where if  $k = 1$ , then  $V_R' = V_R$ , otherwise  $V_R' = V_R \cup V_N''$ , and  $P'$  is constructed as follows:

1. To start with, if  $k' = 1$  then  $P' = \emptyset$ , and if  $k' \geq 2$ , then  $P' = P''$ .

1. This means the following: Let  $A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_{n'}, \pi_{n'}))$  be such a production in  $G'$  corresponding to some production in  $G$ . If some  $i$  is in some  $S_j$ , then  $\pi_i$  is the same permutation as  $\pi_j$ . Also the number of nonterminals in  $\alpha_1$  (and hence in each  $\alpha_i$ ,  $1 \leq i \leq n'$ ) is less than or equal to  $k'$ .



(The idea is to ensure that in case  $k' \geq 2$ , the  $\mu$ -simplicity and the order of  $G'$  will be  $\sigma'$  and  $k'$ , respectively, provided that the productions that will be added to  $P'$ , as we proceed, do not make it an impossibility. Note that if  $k' = 1$ , then  $\xi'_1$  cannot be anything but  $\{2, \dots, n'\}$  and if  $n' = 1$ , then  $\sigma'$  cannot be anything but  $\emptyset$ ; all this, in view of Convention 1.1.1).

2. For each terminating production:  $A \rightarrow (x_1, \dots, x_n)$  in  $P$ , add to  $P'$  the production:  $A \rightarrow (x'_1, \dots, x'_{n'})$ , where if  $j$ ,  $1 \leq j \leq n'$ , is equal to  $i_s$ ,  $1 \leq s \leq n$ , then  $x'_j = x_s$ , otherwise  $x'_j = \epsilon$ .

3. For each nonterminating production  $\psi: A \rightarrow ((\alpha_1, \pi_1), \dots, (\alpha_n, \pi_n))$ , in  $P$ , we shall add a production  $\psi'$  to  $P'$ , which is constructed as follows:

Let  $B_1, \dots, B_l$ , be the nonterminals in  $\alpha_1$ , in order from the left. The production  $\psi'$  is:  $A \rightarrow ((\alpha'_1, \pi'_1), \dots, (\alpha'_{n'}, \pi'_{n'}))$ , where  $\alpha'_1, \dots, \alpha'_{n'}$ , and  $\pi'_1, \dots, \pi'_{n'}$ , are determined using the following steps:

(i) Let  $X = \{j_1, \dots, j_t\}$ , such that  $X = \{1, \dots, n'\} - \xi'_1$ , and the elements of  $X$  are in natural order.

(Notice that at this stage  $X$  will certainly contain 1).

Let  $Y = \{1\} \cup \xi'_1$ . Go to (iv).



(ii) Is  $X$  empty? If it is, then the construction of  $\psi'$  from  $\psi$  is complete; otherwise,

(iii) Let  $i_f$  be the first element in  $X$ . Let  $Y = (X \cap \xi'_{i_f}) \cup \{i_f\}$ .

(From  $s_4$  (this statement appears later in the proof), we have that if some  $i$  is in  $\xi'_{i_f}$ , then it must be in  $X$  as it stands at this stage. Thus, at this stage  $Y$  contains  $i_f$  and all  $i$  such that  $i$  is in  $\xi'_{i_f}$ ).

(iv) Let  $X \leftarrow X - Y$ . Arrange the elements of  $X$  such that they are in natural order.

(v) Test if  $Y \cap \{i_1, \dots, i_n\}$  is empty. If it is empty, then go to (vii); otherwise,

(vi) Let  $t_1, \dots, t_s, s \geq 1, 1 \leq t_1 < \dots < t_s \leq n$ , be such that  $i_{t_1}, \dots, i_{t_s}$ , are in  $Y$ . Then,

(a) For all  $m$ ,  $1 \leq m \leq s$ , let  $\alpha'_{i_{tm}} = \alpha_{tm}$ , and  $\pi'_{i_{tm}} = \pi$

(Notice that  $i_1$  need not be necessarily equal to 1. In such a case,  $\pi'_1$  need not be an identity permutation, through the construction, that we are giving.)

Also, we observe the following statement which we name  $s_1$ .

$s_1$ : It cannot be that for some  $t_p$ , and  $t_q$ ,  $1 \leq p, q \leq s$ ,  $\pi'_{tp}$  is different from  $\pi'_{tq}$ .



Suppose that the statement  $s_1$  is not true, then it follows that  $tq$  is not in  $\xi_{tp}$  and therefore  $i_{tq}$  cannot be in  $\xi'_{i_{tp}}$ , because  $P^L(\sigma', \sigma) = 1$ . If  $i_{tp} = i_f$ , then the contradiction is immediate because then by the very construction of  $Y$ ,  $i_{tq}$  is in  $\xi'_{i_f}$ . Otherwise, the contradiction follows in view of Convention 1.1.1).

(b) For all  $m$  in  $Y - \{i_{t_1}, \dots, i_{t_s}\}$ , let  $\alpha'_m = B \hat{\kappa}_{t_1}(1) \dots B \hat{\kappa}_{t_1}(l)$ , and let  $\kappa'_m = \kappa_{t_1}$ .

(Note that by statement  $s_1$  above,  $\kappa_{t_1} = \dots = \kappa_{t_s}$ ).

Go to (ii).

(vii) For all  $m$  in  $Y$ , let  $\alpha'_m = B_1 \dots B_l$  and let  $\kappa'_m$  be an identity permutation.

(viii) Go to (ii).

Now, we shall show that  $L(G') = L(G)$  and  $G'$  is an  $(n', \sigma', k')$ -CG.

$L(G') = L(G)$ , because  $V''_N \cap V_N = \emptyset$ , and in view of the steps<sup>1</sup> 2, 3(v), 3(vi), and 3(vii). The order of  $G'$  is  $k'$  in view of step 1. Now, the only thing that needs to be proved is that the  $\mathcal{U}$ -simplicity of  $G'$  is  $\sigma'$ . In view of step 1, it suffices to prove that if as a result of step 3, a production:  $\chi': A \rightarrow ((\alpha'_1, \kappa'_1), \dots, (\alpha'_{n'}, \kappa'_{n'}))$ , is added,

---

1. Steps refer to those of the algorithm.



Suppose that the statement  $s_1$  is not true, then it follows that  $tq$  is not in  $\xi_{tp}$  and therefore  $i_{tq}$  cannot be in  $\xi'_{i_{tp}}$ , because  $P^L(\sigma', \sigma) = 1$ . If  $i_{tp} = i_f$ , then the contradiction is immediate because then by the very construction of  $Y$ ,  $i_{tq}$  is in  $\xi'_{i_f}$ . Otherwise, the contradiction follows in view of Convention 1.1.1).

(b) For all  $m$  in  $Y - \{i_{t_1}, \dots, i_{t_s}\}$ , let  $\alpha'_m = B \hat{\pi}_{t_1(1)} \dots B \hat{\pi}_{t_1(l)}$ , and let  $\pi'_m = \pi_{t_1}$ .

(Note that by statement  $s_1$  above,  $\pi_{t_1} = \dots = \pi_{t_s}$ ).

Go to (ii).

(vii) For all  $m$  in  $Y$ , let  $\alpha'_m = B_1 \dots B_l$  and let  $\pi'_m$  be an identity permutation.

(viii) Go to (ii).

Now, we shall show that  $L(G') = L(G)$  and  $G'$  is an  $(n', \sigma', k')$ -CG.

$L(G') = L(G)$ , because  $V_N'' \cap V_N = \emptyset$ , and in view of the steps<sup>1</sup> 2, 3(v), 3(vi), and 3(vii). The order of  $G'$  is  $k'$  in view of step 1. Now, the only thing that needs to be proved is that the  $\mathcal{U}$ -simplicity of  $G'$  is  $\sigma'$ . In view of step 1, it suffices to prove that if as a result of step 3, a production:  $\psi': A \rightarrow ((\alpha'_1, \pi'_1), \dots, (\alpha'_{n'}, \pi'_{n'}))$ , is added,

1. Steps refer to those of the algorithm.



then for no  $q$  in  $\xi'_p$ ,  $1 \leq p \leq n' - 1$ ,  $\pi'_q$  is different from  $\pi'_p$ . This we prove, by proving a sequence of statements.

In view of Convention 1.1.1, the statements  $s_2$  and  $s_3$  follow directly (see the proof of statements  $S_1$  and  $S_2$ , in the proof of Theorem 1.1.1):

$s_2$ : If some  $j$  is in  $\xi'_m$ ,  $1 \leq m \leq n' - 1$ , then

$$\xi'_j = \xi'_m - \{m+1, \dots, j\}.$$

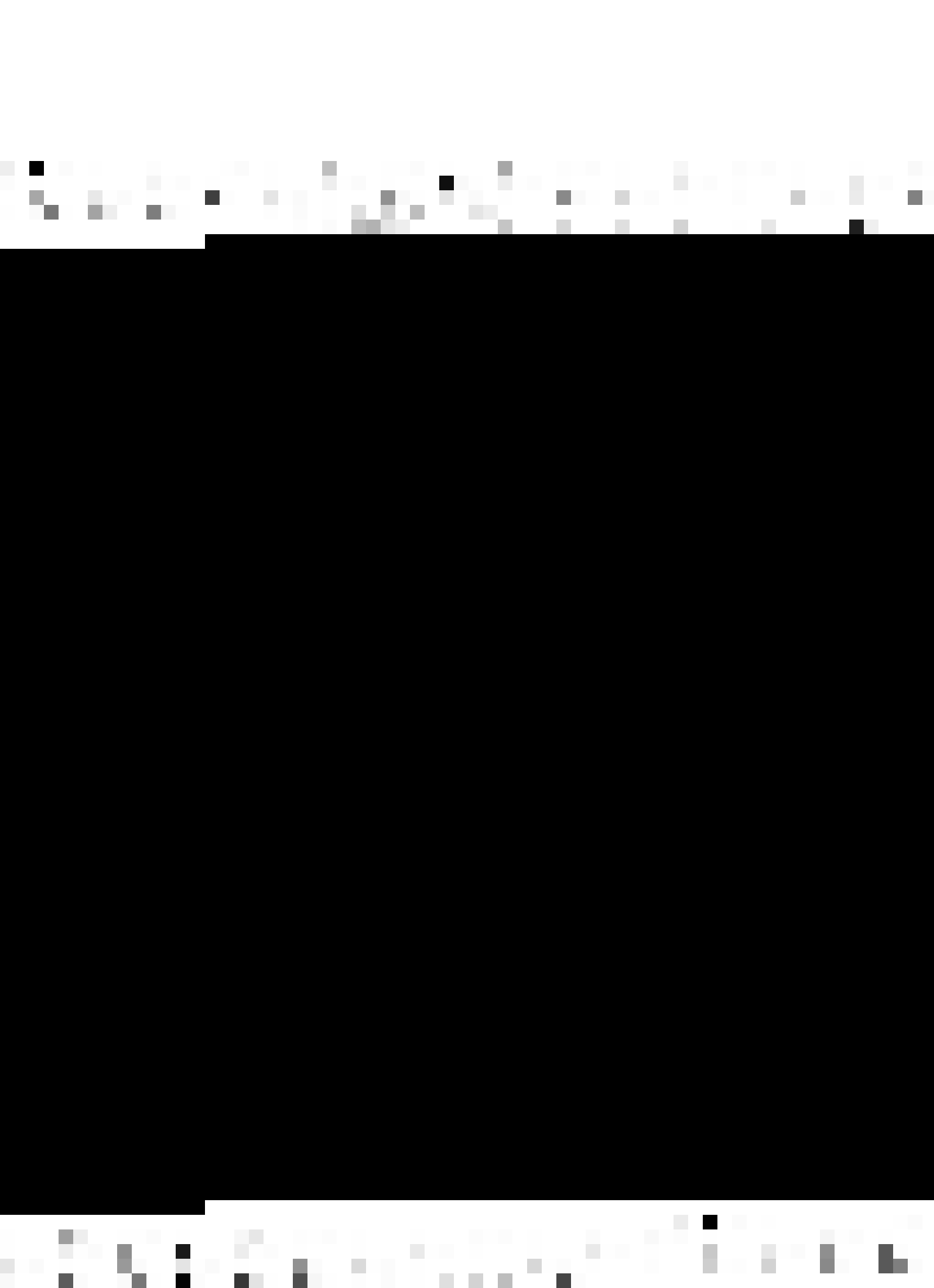
$s_3$ : If at any stage of iteration, some  $j$  is in  $\xi'_{i_f}$ , then  $j$  cannot be in  $\xi'_1$ , and hence  $j$  has to be in  $X$  as defined in step 3(i).

$s_4$ : It cannot be that at any stage of iteration, in step 3(iii) some  $j$  is in  $\xi'_{i_f}$  but is not in  $X$  as it stands in that step.

Suppose this ( $s_4$ ) is not the case, then in view of  $s_3$  and nature of steps 3(iii) through 3(vii), it follows that for some  $m$ ,  $i_f$  is not in  $\xi'_m$  but  $j$  is in  $\xi'_m$ . This leads to a contradiction in view of Convention 1.1.1.

$s_5$ : For all  $j$  in  $\xi'_1$ ,  $\pi'_j = \pi'_1$ .

In the very first iteration,  $Y = \{1\} \cup \xi'_1$ .  $s_5$  now follow in view of nature of steps 3(v) through 3(vii), and statement  $s_1$ .



$s_6$ : Let some  $j$  in  $\{2, \dots, n'\}$ , be not in  $\xi'_1$  and let  $\xi'_j = \{m_1, \dots, m_s\}$ . Then  $\kappa'_j = \kappa'_{m_1} = \dots = \kappa'_{m_s}$ .

This is because at a certain stage of iteration, either  $j = i_f$  or  $j$  is in  $\xi'_{i_f}$ , for some  $i_f$ , in step 3(iii). In the former case,  $s_6$  follows in view of nature of steps 3(v), 3(vi), 3(vii) and statement  $s_4$ . In the latter case, it follows in view of the statements  $s_2$ ,  $s_1$ , and nature of steps 3(v) through 3(vii).

The result follows now in view of the statements  $s_2$ ,  $s_5$ , and  $s_6$ .

Q.E.D.

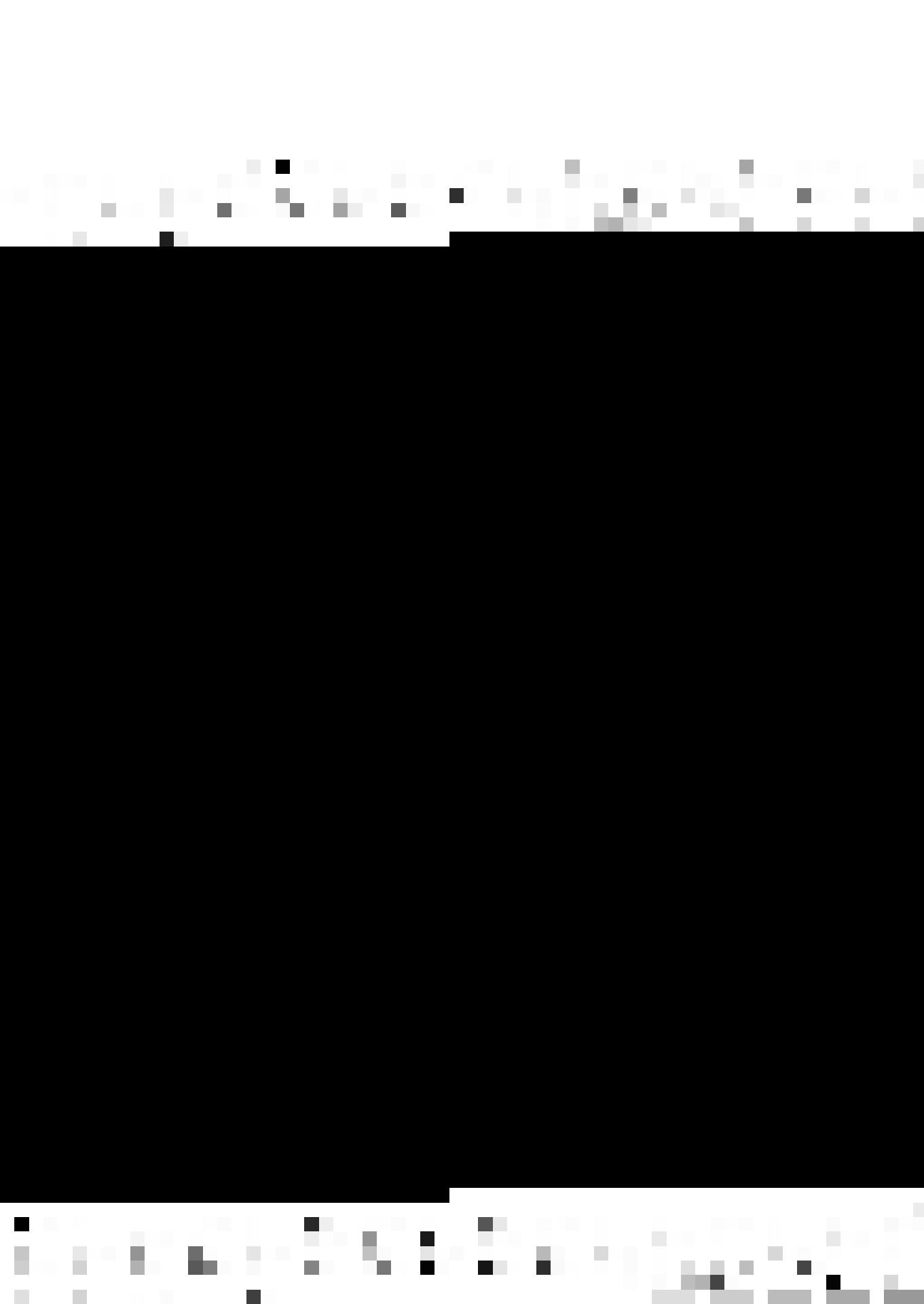
Cor. 2.2.1.1:

Let  $\sigma' = \xi'_1, \dots, \xi'_{n-1}$ , be the  $\mu$ -simplicity of some  $(n, k')$ CG and let  $\sigma = \xi_1, \dots, \xi_{n-1}$ , be the  $\mu$ -simplicity of some  $(n, k)$ CG and  $k' \geq k$ . Let  $\xi'_j \subset \xi_j$ ,  $1 \leq j \leq n-1$ . Then,

$$\mathcal{C}_{(n, \sigma', k')}^L \supset \mathcal{C}_{(n, \sigma, k)}^L$$

\*\*\*

We know that any context-free language can be generated by a context-free grammar in 'Chomsky Normal Form' (8). If  $\sigma'$  is the  $\mu$ -simplicity of some  $(1, 2)$ CG and  $\sigma$  is the  $\mu$ -simplicity of some  $(1, 1)$ CG (of course,  $\sigma' = \sigma = \emptyset$ ), then clearly  $P_L^L(\sigma', \sigma) = 1$ . Thus, we have:



Cor. 2.2.1.2:

If  $L$  is a context-free language, then it can be generated by some  $(1, \emptyset, 2)$  CG.

Cor. 2.2.1.3:

- (a)  $\mathcal{C}_{(n', \sigma')}^L \supset \mathcal{C}_{(n, \sigma)}^L$ , where  $n' \geq n$ , and  $\rho^L(\sigma', \sigma) = 1$
- (b)  $\mathcal{C}_{(n', k')}^L \supset \mathcal{C}_{(n, k)}^L$ , where  $n' \geq n$ , and  $k' \geq k$ .

\*\*\*

In view of Cor. 2.2.1.1. and Theorem 1.4.2, we have:

Cor. 2.2.1.4:

Any  $(n, k)$  SCL can be generated by an  $(n, 2)$  SCG, in simple normal form.

Cor. 2.2.1.5:

- (a)  $\mathcal{C}_{n'}^L \supset \mathcal{C}_n^L$ ,  $n' \geq n$ ,
- (b)  $\mathcal{S}_{n'}^L \supset \mathcal{S}_n^L$ ,  $n' \geq n$ .

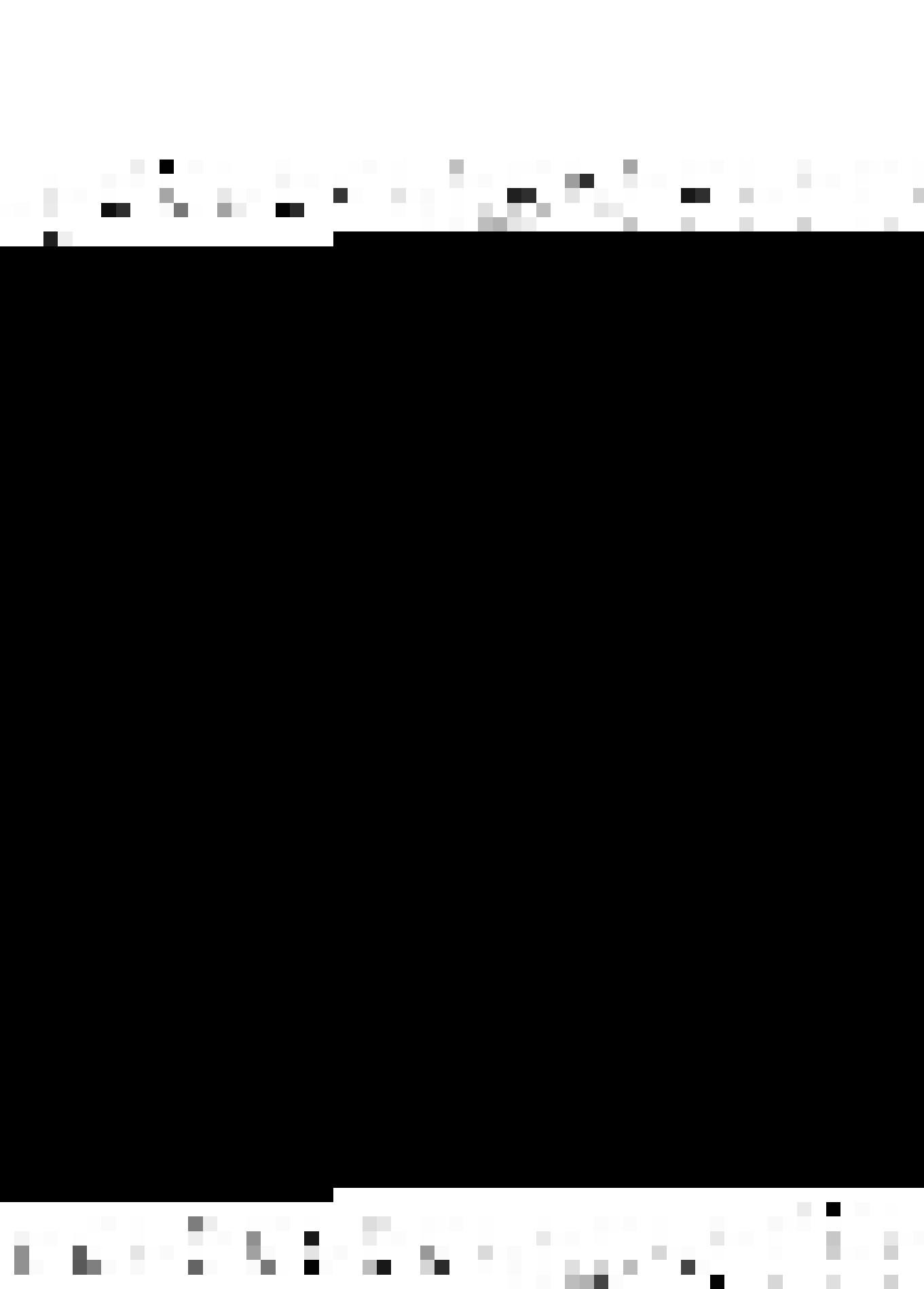
\*\*\*

We shall illustrate the proof of the above theorem (and hence also the algorithm given in the proof) through the following example.

Example 2.2.1:

Consider the language

$$L = \left\{ a_1^{n_1} b_1^{n_1} a_2^{n_2} b_2^{n_2} a_1^{n_1} b_1^{n_1} a_2^{n_2} b_2^{n_2} a_2^{n_2} b_2^{n_2} a_1^{n_1} b_1^{n_1} \mid n_1, n_2 \geq 1 \right\},$$



which is an  $(n, \sigma, k)$ CL, where  $n = 3$ ,  $k = 2$ ,  $\sigma = \xi_1, \xi_2$ ,  
 $\xi_1 = \{2\}$ ,  $\xi_2 = \emptyset$ , and which is generated by the  
 $(n, \sigma, k)$ CG:

$$G = (\{S, A, B\}, \{a_1, b_1, a_2, b_2\}, P, (S)^3),$$

where  $P$  comprises of the productions:

1.  $S \rightarrow (AB, AB, (BA, [2, 1])),$
2.  $A \rightarrow (a_1 Ab_1)^3,$
3.  $B \rightarrow (a_2 Ab_2)^3,$
4.  $A \rightarrow (a_1 b_1)^3, \text{ and}$
5.  $B \rightarrow (a_2 b_2)^3.$

Let us show that  $L$  is also an  $(n', \sigma', k')$ CL, where  $n' = 4$ ,  $\sigma' = \xi'_1, \xi'_2, \xi'_3$ , such that  $\xi'_1 = \{4\}$ ,  $\xi'_2 = \{3\}$ ,  $\xi'_3 = \emptyset$ , and  $k' = 3$ .

We know that  $\rho^L(\sigma', \sigma) = 1$ , because there exist integers  $i_1 = 2$ ,  $i_2 = 3$ , and  $i_3 = 4$ , such that for every  $j$ ,  $1 \leq j \leq 2$ , if  $s$  in  $\{j+1, \dots, 3\}$  is not in  $\xi_j$ , then  $i_s$  is not in  $\xi'_{i_j}$  (see Example 2.1.2).

We shall use the algorithm given in the proof of the above theorem to construct an  $(n', \sigma', k')$ CG  $G'$  such that  $L(G') = L$ . (In this construction, we shall be referring to the steps of this algorithm).



Let  $G'' = (\{A_1, A_2, A_3, B_1, B_2, S_1\}, \emptyset, P'', (S_1)^4)$ ,  
be the sample  $(n', \sigma', k')$  CG, where  $P''$  comprises of the productions:

$$1. \quad S_1 \rightarrow (A_1 A_2 A_3)^4, \text{ and}$$

$$2. \quad S_1 \rightarrow (B_1 B_2, (B_2 B_1, [2, 1]), (B_2 B_1, [2, 1]), B_1 B_2).$$

(See Example 1.1.5).

$$\text{Now, } G' = (V_N', V_T, P', (S)^4),$$

where  $V_N' = \{A_1, A_2, A_3, B_1, B_2, S_1, S, A, B\}$ ,  $V_T = \{a_1, b_1, a_2, b_2\}$ , and  $P'$  is constructed as follows:

(a) To start with,  $P' = P''$ .

(b) Using step 2 of the algorithm, we add the productions:

$A \rightarrow (\epsilon, a_1 b_1, a_1 b_1, a_1 b_1)$ , and  $B \rightarrow (\epsilon, a_2 b_2, a_2 b_2, a_2 b_2)$ , to  $P'$ .

(c) There are three nonterminating productions in  $P$ . Corresponding to each of these productions, we shall add a production in  $P'$ . First let us consider the production:  $S \rightarrow (AB, AB, (BA, [2, 1]))$  in  $P$ .

Corresponding to this production we shall add the production:

$$S \rightarrow ((\alpha'_1, \pi'_1), (\alpha'_2, \pi'_2), (\alpha'_3, \pi'_3), (\alpha'_4, \pi'_4)),$$

where  $\alpha'_1, \alpha'_2, \alpha'_3, \alpha'_4$ , and  $\pi'_1, \pi'_2, \pi'_3, \pi'_4$ , are determined as follows:



$$1. \quad X = \{1, 2, 3\} .$$

$$2. \quad Y = \{1, 4\} . \quad X \leftarrow \{2, 3\} .$$

3. We observe that  $i_3 = 4$  is in Y.

Let  $\alpha'_4 = BA$  and  $\pi'_4 = [2, 1]$ . Let  $\alpha'_1 = BA$  and  $\pi'_1 = [2,$

$$4. \quad X \text{ is still not empty. } Y = \{2, 3\} . \quad X \leftarrow \emptyset .$$

5. We observe that  $i_1 = 2$ , and  $i_2 = 3$ , are in Y.

Let  $\alpha'_2 = AB$ ,  $\pi'_2 = [1, 2]$  and  $\alpha'_3 = AB$ ,  $\pi'_1 = [1, 2]$

6. The specification of the production added is complete, since  $X = \emptyset$ .

It is easy to see that the productions that will be added to  $P'$ , corresponding to the other two nonterminating productions of P will be:  $A \rightarrow (A, a_1Ab_1, a_1Ab_1, a_1Ab_1)$ , and  $B \rightarrow (B, a_2Bb_2, a_2Bb_2, a_2Bb_2)$ .

Thus,  $P'$  comprises of the productions:

$$1. \quad S \rightarrow ((BA, [2, 1]), AB, AB, (BA, [2, 1])),$$

$$2. \quad A \rightarrow (A, a_1Ab_1, a_1Ab_1, a_1Ab_1),$$

$$3. \quad B \rightarrow (B, a_2Bb_2, a_2Bb_2, a_2Bb_2),$$

$$4. \quad A \rightarrow (\epsilon, a_1b_1, a_1b_1, a_1b_1),$$

$$5. \quad B \rightarrow (\epsilon, a_2b_2, a_2b_2, a_2b_2),$$

$$6. \quad S_1 \rightarrow (A_1A_2A_3)^4, \text{ and}$$

$$7. \quad S_1 \rightarrow (B_1B_2, (B_2B_1, [2, 1]), (B_2B_1, [2, 1]), B_1B_2).$$



Clearly,  $L(G') = L$ , and  $G'$  is an  $(4, \{4\}, \{3\}, \emptyset, 3)CG$ .

\*\*\*

We note that in the above example, we can drop production no. 7 of  $P'$  without losing anything. But this need not be the case, in general.

Now, we come to the main results of this section (Theorems 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.7, and 2.2.8) which deal with the structure induced on coupled languages and translations by the degree of the coupled grammars generating them.

The following theorem follows from Lemma 2.2.1 and Theorem 2.2.1:

Theorem 2.2.2:

- (a)  $\mathcal{C}_{(n', \sigma', k')}^L \not\supseteq \mathcal{C}_{(n, \sigma, k)}^L$ , where  $n' > n$ ,  $k' \geq k$ ,  $k' \geq 1$ , and  $\rho^L(\sigma', \sigma) = 1$ .
- (b)  $\mathcal{C}_{(n', k')}^L \not\supseteq \mathcal{C}_{(n, k)}^L$ , where  $n' > n$ ,  $k' \geq k$  and  $k' \geq 1$ .
- (c)  $\mathcal{C}_{(n', \sigma')}^L \not\supseteq \mathcal{C}_{(n, \sigma)}^L$ , where  $n' > n$ , and  $\rho^L(\sigma', \sigma) = 1$ .
- (d)  $\mathcal{C}_{n'}^L \not\supseteq \mathcal{C}_n^L$ , where  $n' > n$ .

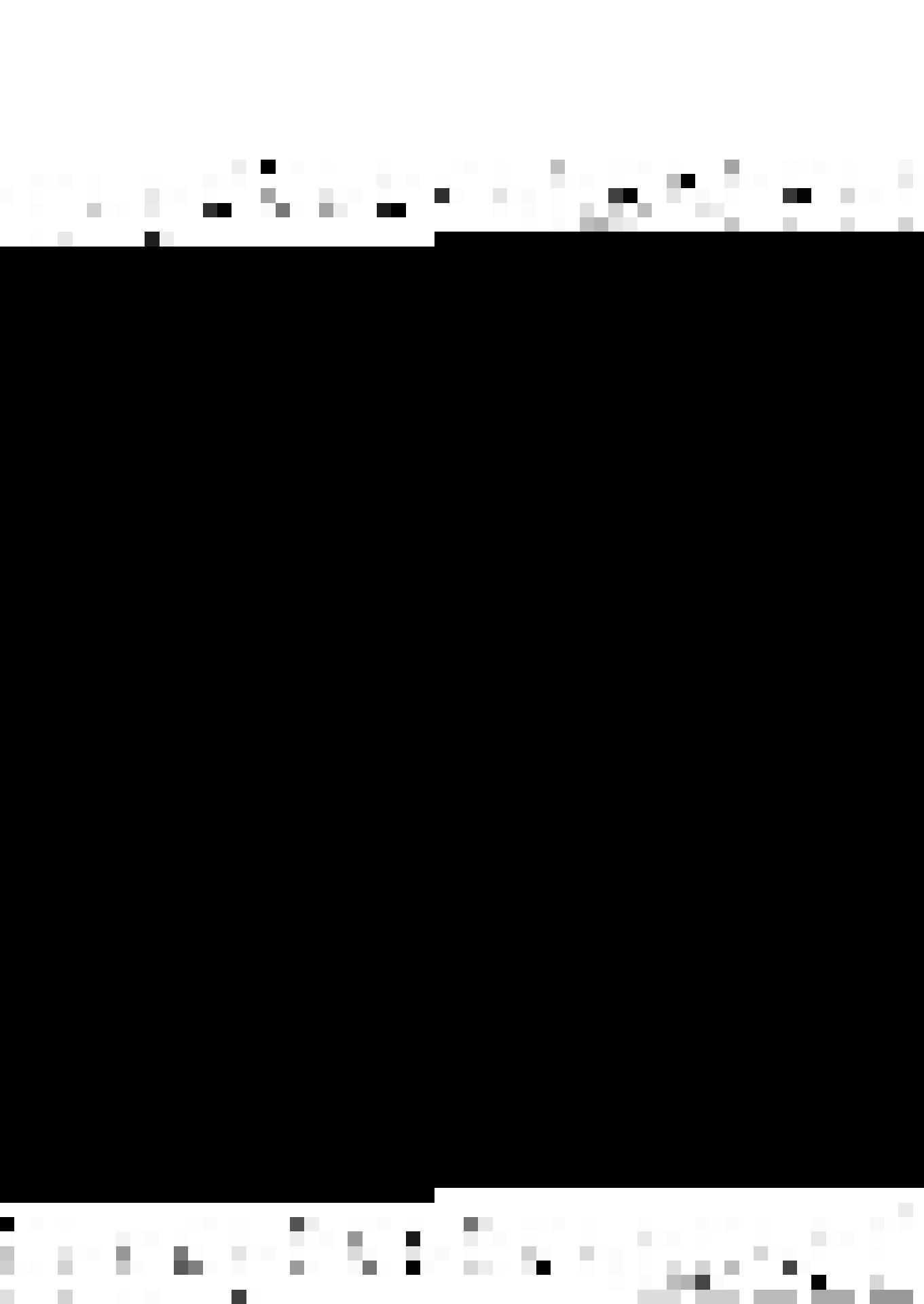
\*\*\*

From Lemma 2.2.2 and Theorem 2.2.1, we have:

Theorem 2.2.3:

- (a)  $\mathcal{S}_{(n', k')}^L \not\supseteq \mathcal{S}_{(n, k)}^L$ , where  $n' > n$ ,  $k' \geq k$  and  $k' \geq 1$ .
- (b)  $\mathcal{S}_{n'}^L \not\supseteq \mathcal{S}_n^L$ , where  $n' > n$ .

\*\*\*



The following theorem follows from Lemma 2.2.1 and Theorem 2.2.1.

Theorem 2.2.4:

Let  $m' > m$ , and  $n > n'$ . Then,

(a) There is a translation in  $\mathcal{C}_{(m', n', \sigma, k)}^T$ ,  $k \geq 1$  which is not in  $\mathcal{C}_{(m, n)}^T$ .

(b) There is a translation in  $\mathcal{C}_{(m, n, \sigma, k)}^T$ ,  $k \geq 1$ , which is not in  $\mathcal{C}_{(m', n')}^T$ .

Cor. 2.2.4.1:

$\mathcal{C}_{(m', n')}^T$  and  $\mathcal{C}_{(m, n)}^T$ , are incomparable, where  $m' > m$ , and  $n > n'$ .

\*\*\*

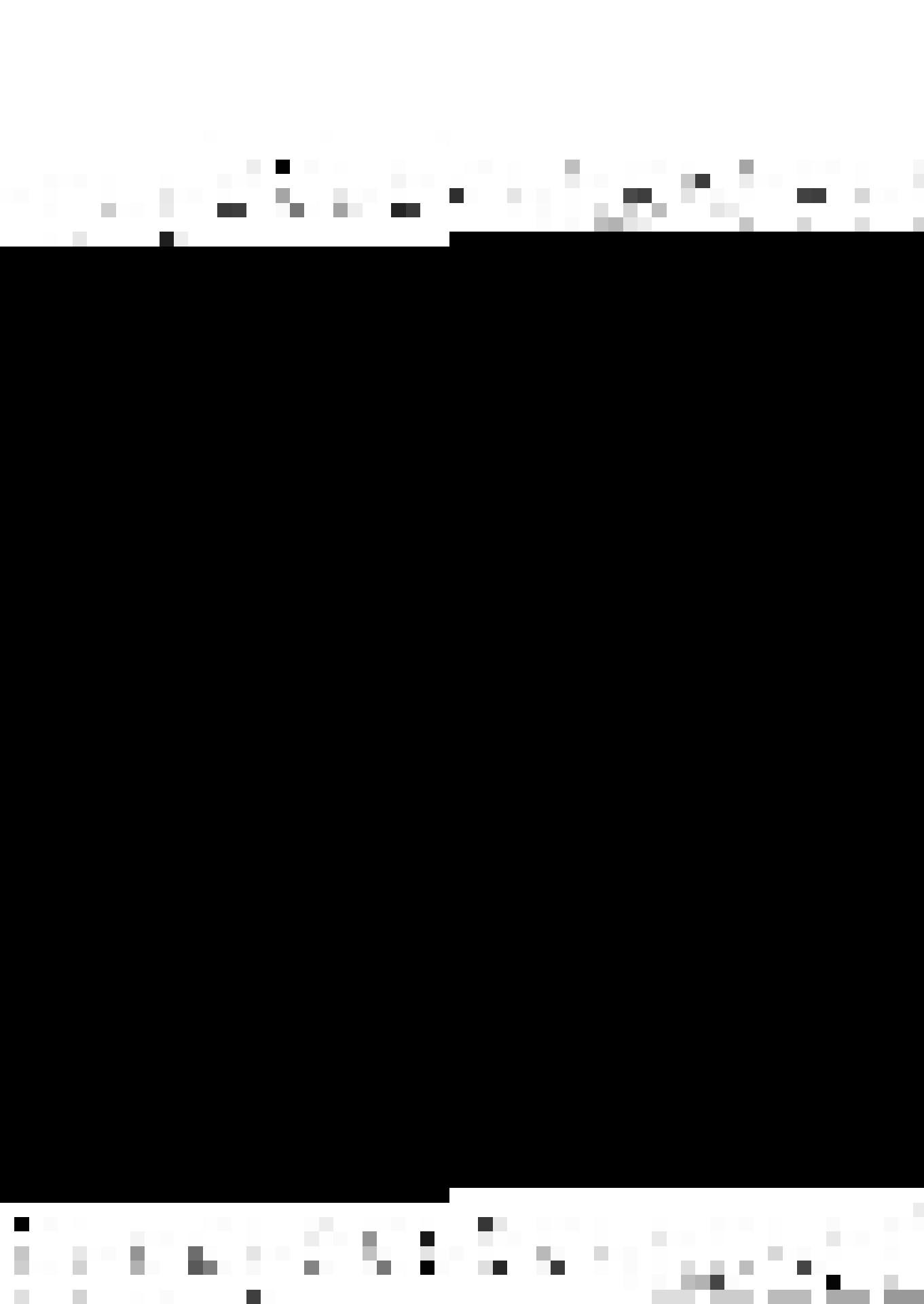
The following theorem follows from Lemma 2.2.2 and Theorem 2.2.1:

Theorem 2.2.5:

(a) There is a translation in  $\mathcal{S}_{(m', n', k')}^T$ ,  $k \geq 1$ , which is not in  $\mathcal{S}_{(m, n)}^T$ , and there is a translation in  $\mathcal{S}_{(m, n, k)}^T$ ,  $k \geq 1$ , which is not in  $\mathcal{S}_{(m', n')}^T$ ,  $m' > m$ ,  $n > n'$ .

(b)  $\mathcal{S}_{(m', n')}^T$  and  $\mathcal{S}_{(m, n)}^T$ , are incomparable, where  $m' > m$ , and  $n > n'$ .

\*\*\*



The proof of the following result is on the same lines as that of Theorem 2.2.1, and is omitted.

Theorem 2.2.6:

$$\mathcal{C}_{(m', n', \sigma', k')}^T \supset \mathcal{C}_{(m, n, \sigma, k)}^T, \text{ where } m' \geq m, n' \geq n, \\ P^T(\sigma', \sigma, m', m) = 1, \text{ and } k' \geq k.$$

Cor. 2.2.6.1:

$$(a) \quad \mathcal{C}_{(m', n', \sigma')}^T \supset \mathcal{C}_{(m, n, \sigma)}^T, \text{ where } m' \geq m, n' \geq n, \text{ and} \\ P^T(\sigma', \sigma, m', m) = 1.$$

$$(b) \quad \mathcal{C}_{(m', n', k')}^T \supset \mathcal{C}_{(m, n, k)}^T, \text{ where } m' \geq m, n' \geq n, \\ \text{and } k' \geq k.$$

\*\*\*

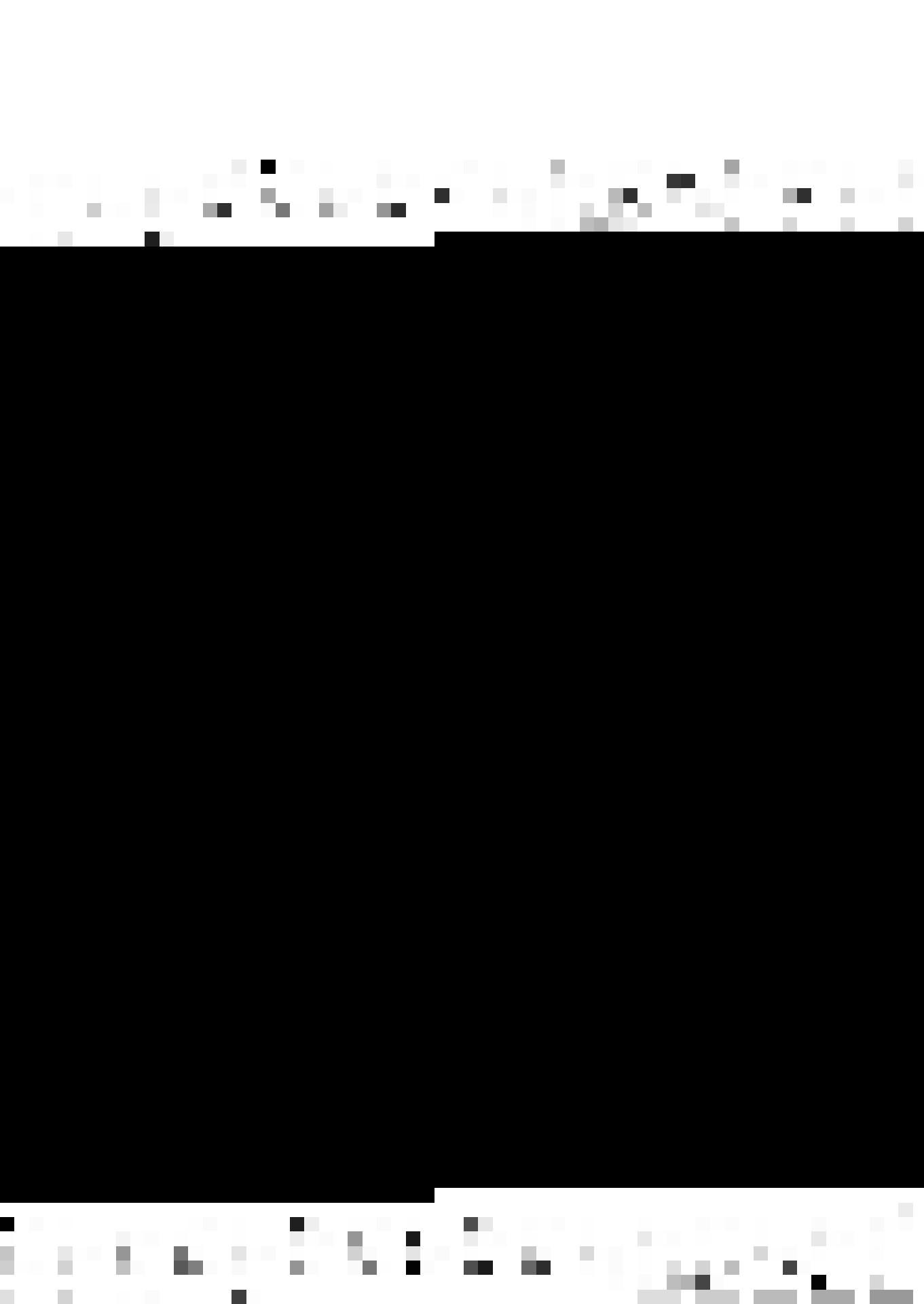
Now, from Theorem 1.2.1, Lemma 2.2.1, and Theorem 2.2.6, the following result follows:

Theorem 2.2.7:

$$(a) \quad \mathcal{C}_{(m', n', \sigma', k')}^T \supset \mathcal{C}_{(m, n, \sigma, k)}^T, \text{ where } m' \geq m, n' \geq n, \\ m' + n' > m + n, k' \geq k, k' \geq 1, \text{ and } P^T(\sigma', \sigma, m', m) = 1.$$

$$(b) \quad \mathcal{C}_{(m', n', k')}^T \supset \mathcal{C}_{(m, n, k)}^T, \text{ where } m' \geq m, n' \geq n, \\ m' + n' > m + n, k' \geq k, \text{ and } k' \geq 1.$$

$$(c) \quad \mathcal{C}_{(m', n', \sigma')}^T \supset \mathcal{C}_{(m, n, \sigma)}^T, \text{ where } m' \geq m, n' \geq n, \\ m' + n' > m + n, \text{ and } P^T(\sigma', \sigma, m', m) = 1.$$



(d)  $\mathcal{C}_{(m', n')}^T \not\supseteq \mathcal{C}_{(m, n)}^T$ , where  $m' \geq m$ ,  $n' \geq n$ , and  $m'+n' > m+n$ .

\*\*\*

Again, from Theorem 1.2.1, Lemma 2.2.2, and Theorem 2.2.6, we have:

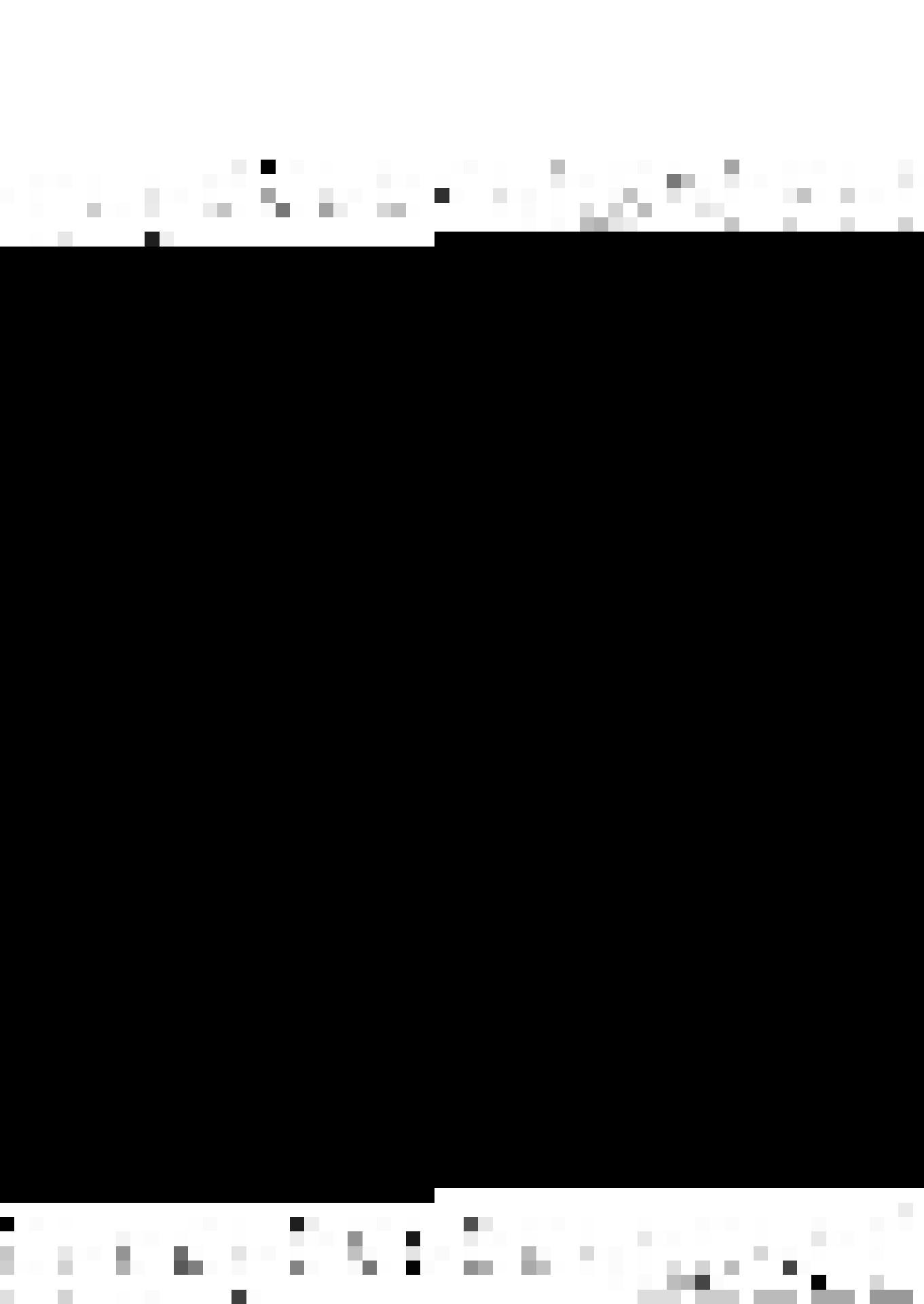
Theorem 2.2.8:

(a)  $\mathcal{S}_{(m', n', k')}^T \not\supseteq \mathcal{S}_{(m, n, k)}^T$ , where  $m' \geq m$ ,  $n' \geq n$ ,  $m'+n' > m+n$ ,  $k' \geq k$ , and  $k' \geq 1$ .

(b)  $\mathcal{S}_{(m', n')}^T \not\supseteq \mathcal{S}_{(m, n)}^T$ , where  $m' \geq m$ ,  $n' \geq n$ , and  $m'+n' > m+n$ .

2.3 STRUCTURE RESULTS II:

Now, we proceed to investigate the structure induced on coupled languages and coupled translations by the  $\mathcal{U}$ -simplicity of the coupled grammars. We shall confine ourselves to proving the required structure results for coupled languages; of course, certain structure results for coupled translations will follow from these. The methodology employed for doing this is to show that there exists a language which is generated by one class of grammars but cannot be generated by another class of grammars. So, before we actually prove these results, we need a class of languages, which we can use to prove the various 'negative' results. Our plan will be to first study the structure of languages in this class, and then, for proving a certain result,



to show that there is a language in this class whose structure is such that it cannot be generated by any grammar in a certain class.

Let us start by introducing the above mentioned class of languages.

For every  $n \geq 2$ ,  $e \geq 1$ , and  $\xi_i$ ,  $1 \leq i \leq n-1$ , a certain subset of  $\{i+1, \dots, n\}$ , we define below the language:  $L_e(n, \xi_1, \dots, \xi_{n-1})$ , in a structured manner; the above mentioned class comprises of all such languages.

$L_e(n, \xi_1, \dots, \xi_{n-1}) = \{A_1 \xi_1 \dots A_{n-1} \xi_{n-1} g_n$ , where for  $s, p, u, j, v, r, t$ ,  $1 \leq s \leq n$ ,  $p = n-1$ ,  $1 \leq u \leq t$ ,  $1 \leq j \leq p$ ,  $2 \leq v \leq n$ ,  $r \geq 1$ , and  $t \geq 1$ ,

$$(i) \quad A_s = h'_{s1} B_{s1} h_{s1} \dots h'_{st} B_{st} h_{st},$$

$$B_{1u} = D_{1u1} D_{1u2} \dots D_{1up},$$

$$D_{1uj} = a_{j1}^{i_{uj1}} b_{j1}^{i_{uj1}} a_{j2}^{i_{uj2}} b_{j2}^{i_{uj2}},$$

$$B_{vu} = D_{vu1} D_{vu2} \dots D_{vup},$$

$$D_{vuj} = a_{vj1}^{k_{vuj1}} b_{vj1}^{k_{vuj1}} a_{vj2}^{k_{vuj2}} b_{vj2}^{k_{vuj2}}.$$

(ii) For  $r \leq e$ ,  $h_{sr} = g_{sr}$ ,  $h'_{sr} = g'_{sr}$ , and for  $r > e$ , if  $r = e \cdot k + r'$ , where  $k \geq 1$  and  $1 \leq r' \leq e$ , then  $h_{sr} = g_{sr}$ ,  $h'_{sr} = g'_{sr}$ .



(iii) If  $j$  in  $\{2, \dots, n-1\}$  is not in  $\xi_1$ , locate the production:  $S \rightarrow (\beta_1, (\beta_2, \kappa'_2), \dots, (\beta_n, \kappa'_n))$  in  $P'$  — the set of productions of the sample  $(n, \xi_1, \dots, \xi_{n-1}, 2)CG$  (see the proof of Theorem 1.1.1), where  $\kappa'_j$  is a non-identity permutation (note that there is one and only one such production).

If  $\kappa'_v$  is an identity permutation, then  $l_{vj1} = j_1$ ,  $l_{vj2} = j_2$ ,  $k_{vuj_1} = uj_1$ ,  $k_{vuj_2} = uj_2$ , and if  $\kappa'_v$  is not an identity permutation, then  $l_{vj1} = j_2$ ,  $l_{vj2} = j_1$ ,  $k_{vuj_1} = uj_2$ , and  $k_{vuj_2} = uj_1$ .

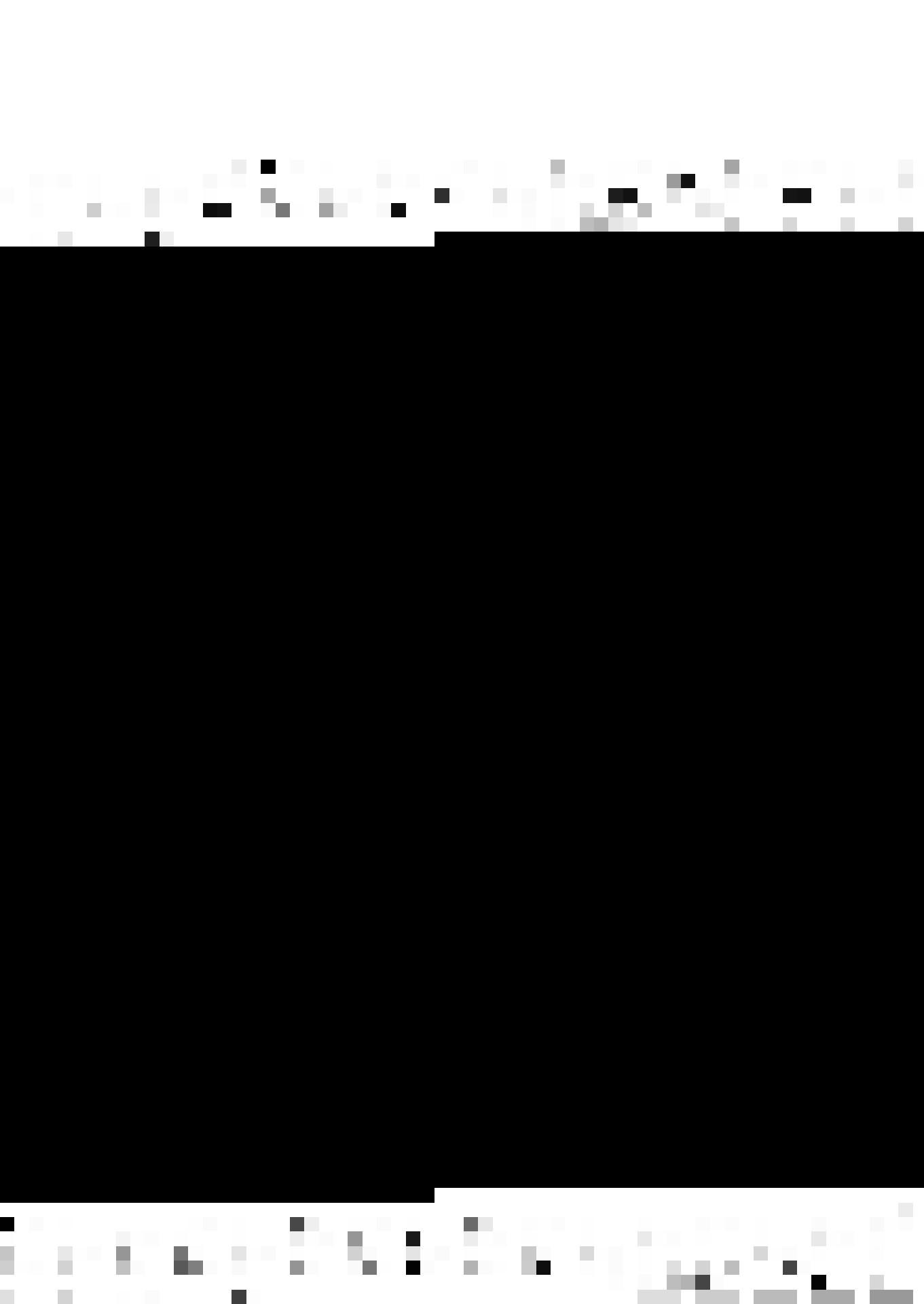
(iv) If  $j = 1$  or  $j$  is in  $\xi_1$  and if  $v$  is in  $\xi_1$ , then  $l_{vj1} = j_1$ ,  $l_{vj2} = j_2$ ,  $k_{vuj_1} = uj_1$ ,  $k_{vuj_2} = uj_2$ , and if  $v$  in  $\{2, \dots, n\}$  is not in  $\xi_1$  then  $l_{vj1} = j_2$ ,  $l_{vj2} = j_1$ ,  $k_{vuj_1} = uj_2$ , and  $k_{vuj_2} = uj_1$

$$\left| i_{uj_1}, i_{uj_2} \geq 1 \right\} .$$

\*\*\*

The definition of the language  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$  is somewhat complex. We will therefore try to get familiar with it through examples and by looking into its structure.

As an example, let us define the language  $L^2_{(3, \emptyset, \emptyset)}$ :



Example 2.3.1:

$$\begin{aligned}
 L^2_{(3,\emptyset,\emptyset)} = \{ & h'_{11} a_{11}^{i_{111}} b_{11}^{i_{111}} a_{12}^{i_{112}} b_{12}^{i_{112}} a_{21}^{i_{121}} b_{21}^{i_{121}} a_{22}^{i_{122}} b_{22}^{i_{122}} h_{11} \dots \\
 & \dots h'_{1t} a_{11}^{i_{t11}} b_{11}^{i_{t11}} a_{12}^{i_{t12}} b_{12}^{i_{t12}} a_{21}^{i_{t21}} b_{21}^{i_{t21}} a_{22}^{i_{t22}} b_{22}^{i_{t22}} h_{1t} g \\
 & h'_{21} a_{12}^{i_{112}} b_{12}^{i_{112}} a_{11}^{i_{111}} b_{11}^{i_{111}} a_{22}^{i_{122}} b_{22}^{i_{122}} a_{21}^{i_{121}} b_{21}^{i_{121}} h_{21} \dots \\
 & \dots h'_{2t} a_{12}^{i_{t12}} b_{12}^{i_{t12}} a_{11}^{i_{t11}} b_{11}^{i_{t11}} a_{22}^{i_{t22}} b_{22}^{i_{t22}} a_{21}^{i_{t21}} b_{21}^{i_{t21}} h_{2t} g \\
 & h'_{31} a_{12}^{i_{112}} b_{12}^{i_{112}} a_{11}^{i_{111}} b_{11}^{i_{111}} a_{21}^{i_{121}} b_{21}^{i_{121}} a_{22}^{i_{122}} b_{22}^{i_{122}} h_{31} \dots \\
 & \dots h'_{3t} a_{12}^{i_{t12}} b_{12}^{i_{t12}} a_{11}^{i_{t11}} b_{11}^{i_{t11}} a_{21}^{i_{t21}} b_{21}^{i_{t21}} a_{22}^{i_{t22}} b_{22}^{i_{t22}} h_{3t} g
 \end{aligned}$$

where for  $r \leq 2$ ,  $h_{ir} = g_{ir}$ ,  $h'_{ir} = g'_{ir}$ , and for  
 $r > 2$ , if  $r = 2 \cdot j + r'$ ,  $j \geq 1$ ,  $1 \leq r' \leq 2$ , then  
 $h_{ir} = g_{ir'}$ ,  $h'_{ir} = g'_{ir'}$ ,  $1 \leq i \leq 3$

$$| i_{uj1}, i_{uj2} \geq 1, 1 \leq u \leq t, t \geq 1, 1 \leq j \leq 2 \}$$

\*\*\*

The following example gives some sentences of the language  
 $(n, \xi_1, \dots, \xi_{n-1})$  for various values of  $e$ ,  $n$ ,  $\xi_1, \dots, \xi_{n-1}$ .

Example 2.3.2:

- (a)  $z = \xi'_{11} a_{11} b_{11} a_{12} b_{12} \xi_{11} \xi'_{12} a_{11} b_{11} a_{12} b_{12} g_{11} g'_{12} a_{11} b_{11} a_{12} b_{12}$   
 $\xi_{11} \xi'_{12} a_{12} b_{12} a_{11} b_{11} \xi_{21} \xi'_{22} a_{12} b_{12} a_{11} b_{11} g_{22} g'_{21} a_{12} b_{12}$   
 $a_{11} b_{11} g_{21} g_2$ , is a sentence in the language  $L^2_{(2,\emptyset)}$ .

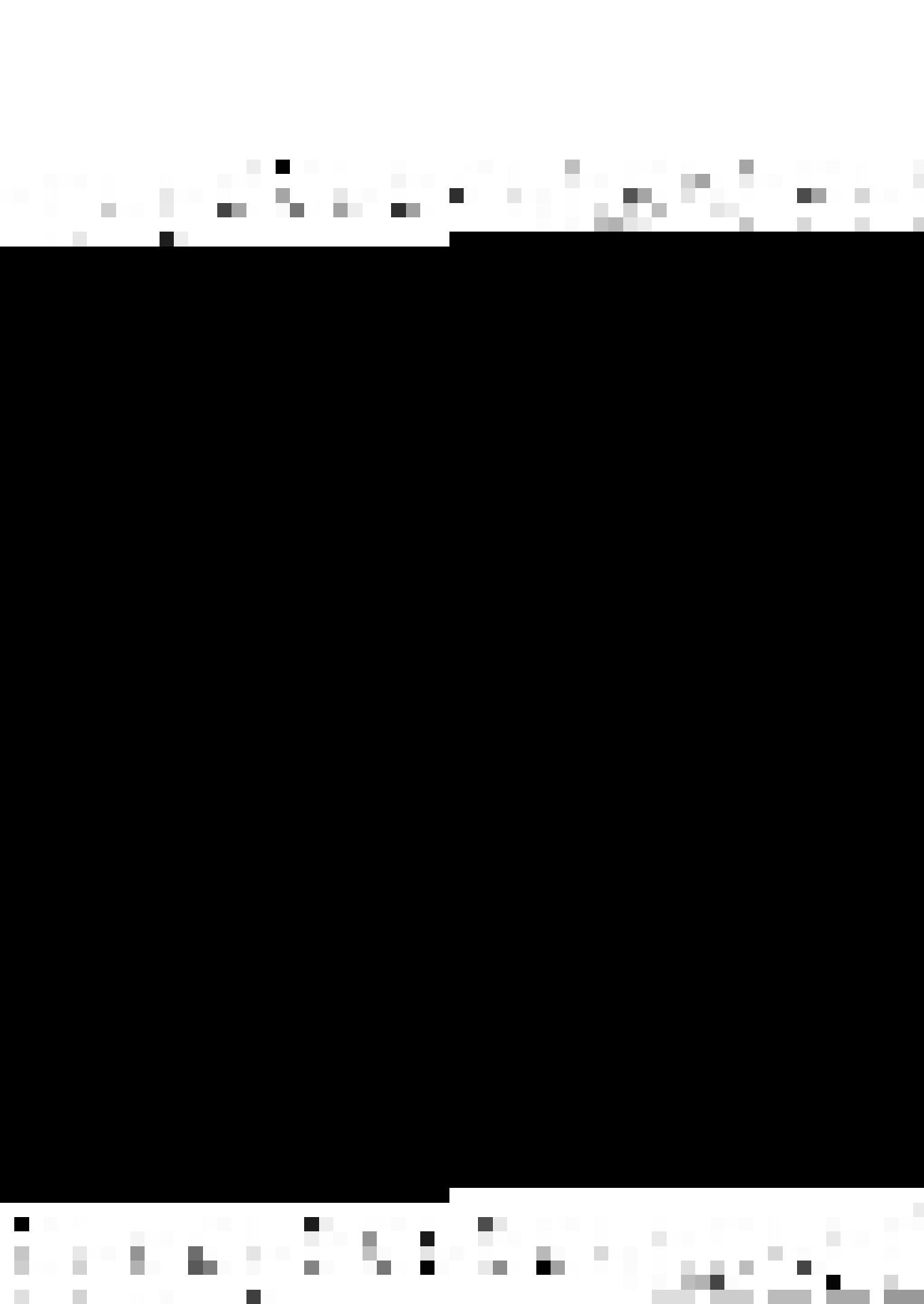


(b)  $z = g'_{11}a_{11}b_{11}a_{12}a_{12}b_{12}b_{12}g_{11}g'_{12}a_{11}a_{11}b_{11}b_{11}a_{12}b_{12}g_{12}$   
 $g'_{13}a_{11}b_{11}a_{12}b_{12}g_{13}g_1g'_{21}a_{12}a_{12}b_{12}b_{12}a_{11}b_{11}g_{21}$   
 $g'_{22}a_{12}b_{12}a_{11}b_{11}b_{11}g_{22}g'_{23}a_{12}b_{12}a_{11}b_{11}g_{23}g_2,$   
is a sentence in the language  $L^3_{(2, \emptyset)}$ .

(c)  $z = g'_{11}a_{11}b_{11}a_{12}b_{12}a_{21}b_{21}a_{22}b_{22}g_{11}g'_{12}a_{11}a_{11}b_{11}b_{11}$   
 $a_{12}b_{12}a_{21}a_{21}b_{21}a_{22}b_{22}g_{12}g_1g'_{21}a_{12}b_{12}a_{11}b_{11}$   
 $a_{22}b_{22}a_{21}b_{21}g_{21}g'_{22}a_{12}b_{12}a_{11}a_{11}b_{11}a_{22}b_{22}$   
 $a_{21}a_{21}b_{21}b_{21}g_{22}g'_{31}a_{11}b_{11}a_{12}b_{12}a_{21}b_{21}a_{22}b_{22}g_{31}$   
 $g'_{32}a_{11}a_{11}b_{11}b_{11}a_{12}b_{12}a_{21}a_{21}b_{21}a_{22}b_{22}g_{32}g_3,$   
is a sentence in the language  $L^3_{(3, \{3\}, \emptyset)}$ .

\*\*\*

Consider some sentence  $z$  in the language  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$ . We observe that every distinct symbol  $a$  in  $X = \{a_{11}, b_{11}, a_{12}, b_{12}, \dots, a_{(n-1)1}, b_{(n-1)1}, a_{(n-1)2}, b_{(n-1)2}\}$  occurs in  $n.t$ ,  $t \geq 1$ , different 'segments'; we shall call these segments as the segments of a and shall call  $z$  as an  $(n.t)$ -segment sentence. We will say that a certain occurrence of a certain symbol  $a$  in  $z$  is of type  $(i,j)$ ,  $1 \leq i \leq n$ ,  $1 \leq j \leq t$ , if it occurs in the  $((i-1).t+j)$ -th segment of  $a$ , in order from the left. Let  $z$  be written as  $xu_{ij}y$ , such that  $u_{ij}$  contains all the type  $(i,j)$  occurrences of all the symbols in  $X$ , it does not contain occurrence of a symbol which is not in  $X$ , and it does not contain an occurrence of a symbol in  $X$  which is of a type



different from  $(i,j)$ , then we shall call  $u_{ij}$  as the  $(i,j)$ -th composite segment of  $z$ . Notice that every composite segment in  $z$  is flanked on the left side by  $g'$ 's and on the right side by  $g$ 's, with certain double subscripts.

We shall illustrate the above notations through the following example:

Example 2.3.3:

We know that  $z = g'_1 a_{11} b_{11} a_{12} b_{12} g_{11} g'_{12} a_{11} b_{11} a_{12} b_{12} g_{12} g'_1 a_{11} b_{11} a_{12} b_{12} g_{11} g'_{21} a_{12} b_{12} a_{11} b_{11} g_{21} g'_{22} a_{12} b_{12} a_{11} b_{11} g_{22} g'_{21} a_{12} b_{12} a_{11} b_{11} g_{21} g_2$ , is a sentence in the language  $L^2_{(2,\emptyset)}$  (see Example 2.3.2(a)). Observe that  $z$  is a  $(2,3)$ -segment sentence. To illustrate the segments of  $a_{11}$ , we rewrite  $z$  with vertical lines drawn at the boundaries of the segments of  $a_{11}$ , as follows:

$$\begin{aligned} z &= g'_1 | a_{11} | b_{11} a_{12} b_{12} g_{11} g'_{12} | a_{11} | b_{11} a_{12} b_{12} g_{12} \\ &\quad g'_1 | a_{11} | b_{11} a_{12} b_{12} g_{11} g'_{21} a_{12} b_{12} | a_{11} | b_{11} g_{21} \\ &\quad g'_{22} a_{12} b_{12} | a_{11} | b_{11} g_{22} g'_{21} a_{12} b_{12} | a_{11} | b_{11} g_{21} g_2. \end{aligned}$$

The occurrence of  $a_{11}$  in the first segment (of  $a_{11}$ ) (in order from the left) is of type  $(1,1)$ , that in the second segment is of type  $(1,2)$ , that in the third segment is of type  $(1,3)$ , that in the fourth segment is of type  $(2,1)$ , and so on.

We observe that there are six composite segments in  $z$ . To illustrate them, we shall again rewrite  $z$ , marking



out each composite segment by drawing vertical lines at its boundaries, as follows:

$$\begin{aligned} z = & \ g'_1 | a_{11} b_{11} a_{12} b_{12} | g_{11} g'_{12} | a_{11} b_{11} a_{12} b_{12} | g_{12} \\ & \ g'_1 | a_{11} b_{11} a_{12} b_{12} | g_{11} g_1 g'_{21} | a_{12} b_{12} a_{11} b_{11} | g_{21} \\ & \ g'_{22} | a_{12} b_{12} a_{11} b_{11} | g_{22} g'_{21} | a_{12} b_{12} a_{11} b_{11} | g_{21} g_2 . \end{aligned}$$

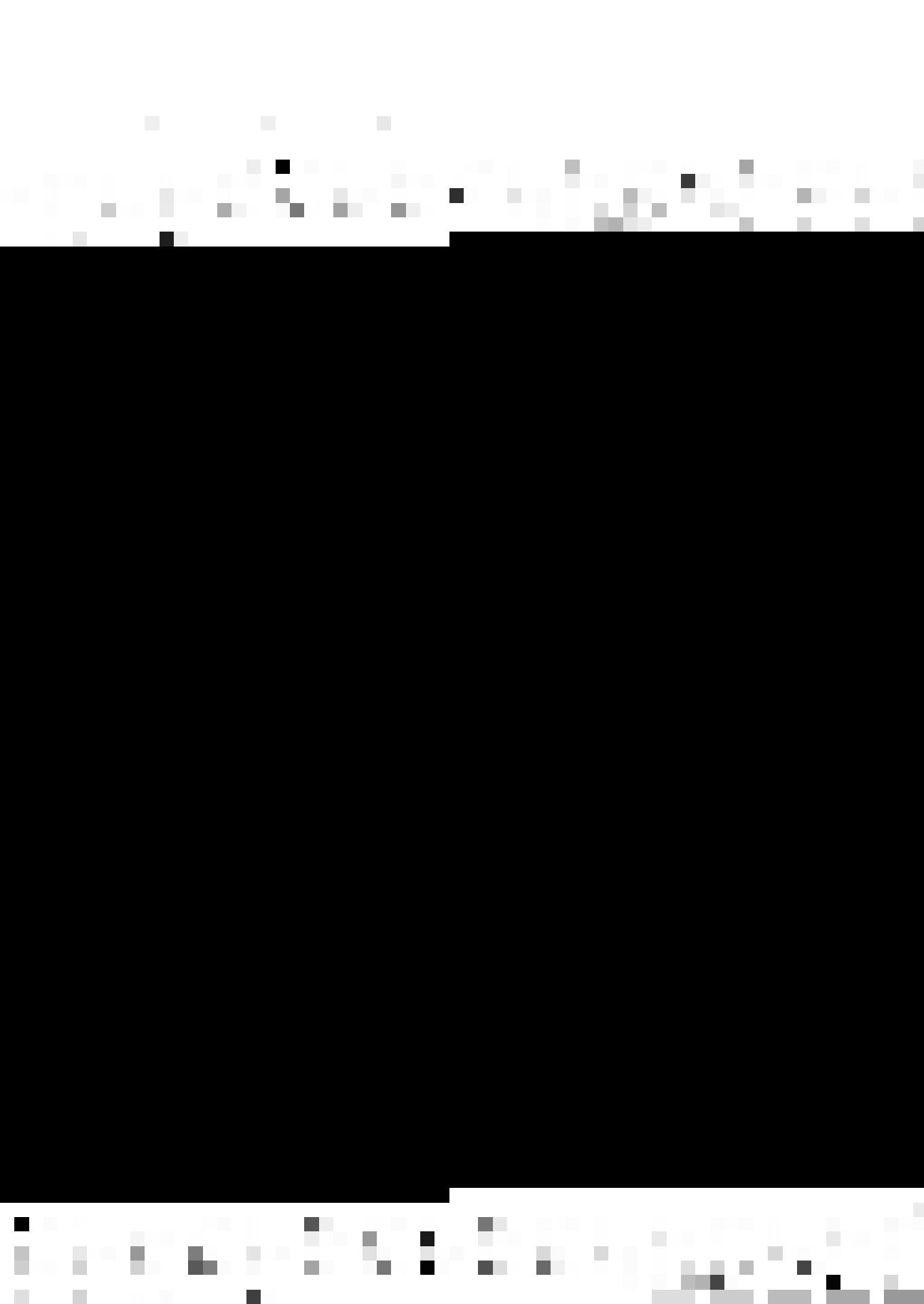
\*\*\*

We shall now state through the following lemma some structural properties which hold true for every sentence in  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$  (which are deduced from the definition of the language  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$ ):

### Lemma 2.3.1:

Let  $z$  be an  $(n, t)$ -segment sentence in  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$ . Let  $X = \{a_{11}, b_{11}, a_{12}, b_{12}, \dots, a_{(n-1)1}, b_{(n-1)1}, a_{(n-1)2}, b_{(n-1)2}\}$ . We have:

- (a) For each  $j$ ,  $1 \leq j \leq t$ , the number of type  $(1, j)$  occurrences of a certain symbol in  $X$ , is the same as the number of its type  $(i, j)$  occurrences,  $2 \leq i \leq n$ .
- (b) If we consider the symbols  $g'$ 's and  $g$ 's with double subscripts, and  $g$ 's with single subscripts, that occur in  $z$ , we notice that they follow a definite pattern. Let us form a string  $G$  out of them, retaining the order in which they occur in  $z$ . We notice the following:



(i) Let  $t \leq e$ . Then,  $G = g'_1g_{11}\dots g'_{1t}g_{1t}g_1g'_2g_{21}\dots \dots g'_{2t}g_{2t}g_2\dots g'_{n1}g_{n1}\dots g'_{nt}g_{nt}g_n$ .

(ii) Let  $t > e$ . Let  $t = e.j+r$ . Let  $G_i = g'_i g_{i1}\dots \dots g'_{ie}g_{ie}$ . Then<sup>1</sup>,  $G = G^j_1 g'_1 g_{11}\dots g'_{1r}g_{1r}g_1 G^j_2 g'_2 g_{21}\dots g'_{2r}g_{2r}g_2\dots \dots G^j_n g'_n g_{n1}\dots g'_{nr}g_{nr}g_n$ .

(c) The composite segments of  $z$  get easily identified. Each composite segment is flanked on the left side by  $g'$ 's and on the right side by  $g$ 's, with certain double subscripts; the composite segment itself not containing an occurrence of a  $g'$  or a  $g$  (with a double or single subscript).

(d) Each composite segment contains one and only one segment of each of the symbols in  $X$ , in a certain order; the segment of  $a_{ij}$  is, however, immediately to the left of the segment of  $b_{ij}$ ,  $1 \leq i \leq n-1$ ,  $1 \leq j \leq 2$ .

(e) In each composite segment, the segments of  $a_{i1}, b_{i1}, a_{i2}, b_{i2}$  are to the left of the segments of  $a_{(i+1)1}, b_{(i+1)1}, a_{(i+1)2}, b_{(i+1)2}$ ,  $1 \leq i \leq n-2$ .

(f) If the length of a certain segment of some symbol  $a_{ij}$  (respectively,  $b_{ij}$ ) in  $X$ ,  $1 \leq i \leq n-1$ ,  $1 \leq j \leq 2$ , is  $l$ , then there are atleast  $n-1$  more distinct segments of  $a_{ij}$  (respectively,  $b_{ij}$ ) of length  $l$  and atleast  $n$  more segments of  $b_{ij}$  (respectively,  $a_{ij}$ ) of length  $l$ .

1.  $G^j_i$  stands for  $G_i \dots G_i$  (repeated  $j$  times).



(g) Consider some  $s$ ,  $1 \leq s \leq n-1$ . Let  $i$  be in  $\xi_s$ . Then, for each  $j$ ,  $1 \leq j \leq t$ , the segment of  $a_{s1}$  (respectively,  $b_{s1}$ ) is to the left of the segment of  $a_{s2}$  (respectively,  $b_{s2}$ ), in the  $(i,j)$ -th composite segment. Again, let  $i$  in  $\{s+1, \dots, n\}$  be not in  $\xi_s$ . Then, for each  $j$ ,  $1 \leq j \leq t$ , the segment of  $a_{s1}$  (respectively,  $b_{s1}$ ) is to the right of the segment of  $a_{s2}$  (respectively,  $b_{s2}$ ), in the  $(i,j)$ -th composite segment.

\*\*\*

Now, we prove a sequence of two lemmas, which establish a relationship between the structure of the sentences of the language  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$  and the structure of coupled grammars generating them. Actually, the latter lemma viz. Lemma 2.3.3 is the key lemma of this section, and Lemma 2.3.2 is one of the results used to prove it.

### Lemma 2.3.2:

Let  $L$  be the language  $L^e_{(n, \xi_1, \dots, \xi_{n-1})}$ , for some value of  $e, n, \xi_1, \dots, \xi_{n-1}$ , and let it be generated by an  $n'$ CG  $G$ ,  $n' \geq n$ . Let  $T = \langle T_1, \dots, T_{n'} \rangle$ , be an  $n'$ -derivation tree in  $G$  of any  $(n, n')$ -segment sentence  $z$  in  $L$ . Then, for atleast one  $j$ ,  $1 \leq j \leq n'$ , the following is true: For every  $i$ ,  $1 \leq i \leq n$ , there is some  $i_r$ ,  $1 \leq i_r \leq n'$ , such that the  $(i, j)$ -th composite segment in  $z$ , is a substring of  $\text{result}(T_{i_r})$ .



Proof:

Let us suppose to the contrary. This implies that for all  $j$ ,  $1 \leq j \leq n'$ , the following is true: For atleast one  $i_j$ ,  $1 \leq i_j \leq n$ ,  $z$  can be written as  $u_j c_j c'_j v_j$ , where  $c_j, c'_j$  are strings containing type  $(i_j, j)$  occurrences of symbols,  $c_j$  is a substring of  $\text{result}(T_{r_j})$ , and  $c'_j$  is a substring of  $\text{result}(T_{r'_j})$ ,  $1 \leq r_j, r'_j \leq n'$ ,  $r_j \neq r'_j$ .

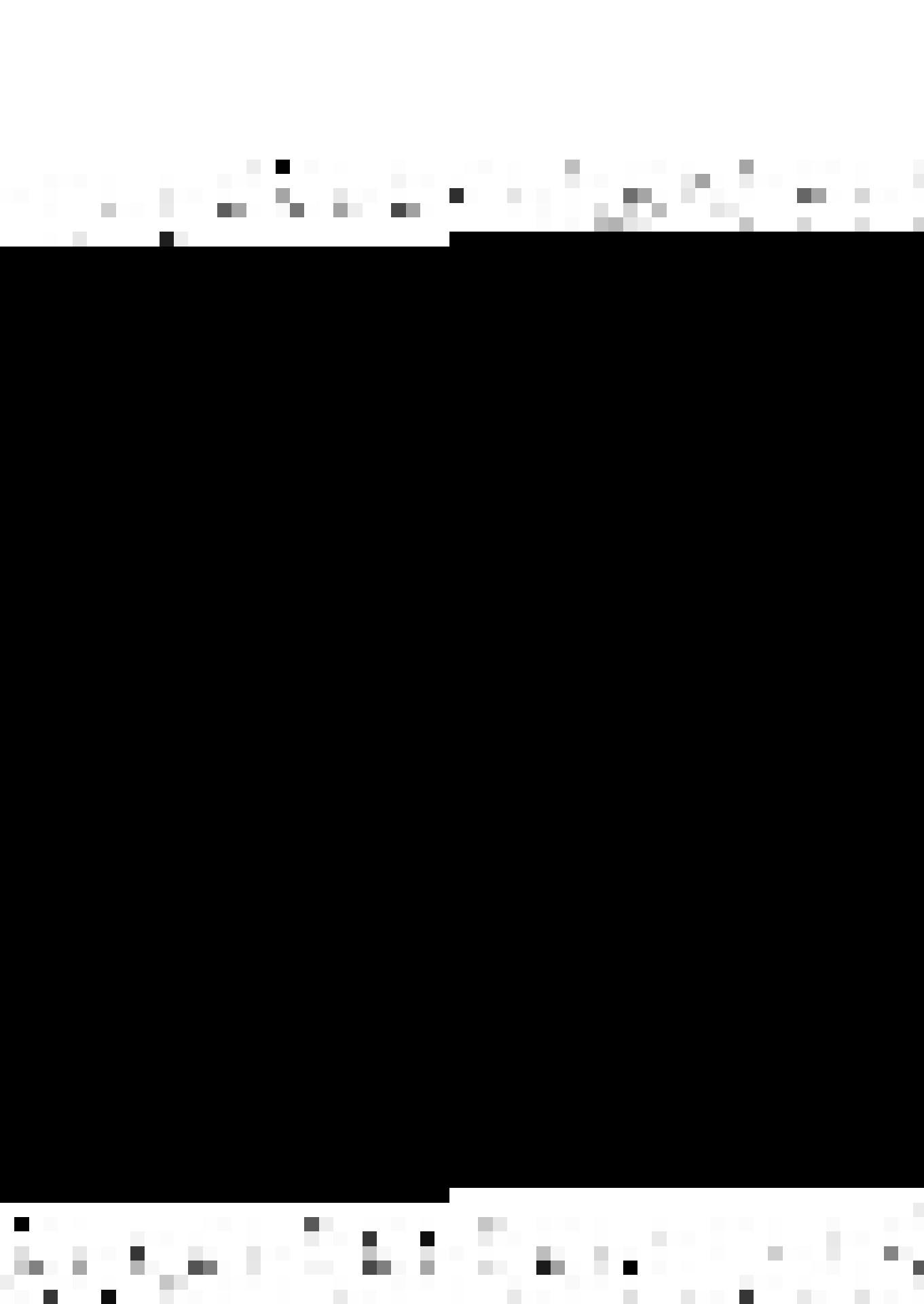
Let us 'partition'  $z$  at different points such that the occurrence of the symbol immediately to the left of a partition  $t$  is in, say,  $\text{result}(T_t)$  and the occurrence of the symbol immediately to the right of  $t$  is in  $\text{result}(T_{t'})$ ,  $t \neq t'$ . It follows from above that we can partition  $z$  at atleast  $n'$  different points, which implies that there are atleast  $n'+1$  trees in  $T$ , which is a contradiction.

Hence the result.

Q.E.D.

Notice that in the above proof, the definitions of composite segments and segments of a sentence in the language  $L^e(n, \xi_1, \dots, \xi_{n-1})$ , do not play any role beyond specifying  $n$  nonempty mutually disjoint substrings of  $z$ , and  $n'$  mutually disjoint substrings of each of them.

Now, we come to the following important lemma. The lemma essentially says certain things about the structure of the  $n'$ -derivation tree in an  $n'CG G$  in normal form , of



some sentence  $z$  in  $L = L_{(n, \xi_1, \dots, \xi_{n-1})}^e$ ,  $n' \geq n$ , if at all  $L$  can be generated by  $G$ .

### Lemma 2.3.3:

Let  $L$  be the language  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$ , for some  $n$ ,  $n'$ ,  $\xi_1, \dots, \xi_{n-1}$ ,  $n' \geq n$ . Let it be generated by an  $n'CG$   $G$  in normal form. There exists a sentence  $z$  in  $L$  having the following property: Let  $T = \langle T_1, \dots, T_{n'} \rangle$  be an  $n'$ -derivation tree of  $z$  in  $G$ . For some  $i_1, \dots, i_n$ ,  $1 \leq i_1 < \dots < i_n \leq n'$ , there are  $((M_{i1}, N_{i1}), (M_{i2}, N_{i2}))$  - mutually distinct  $n'$ -paths,  $1 \leq i \leq n-1$ , in  $T$ , having the property that if the  $(i_c, \dots, i_n)$ -distribution of the  $((M_{c1}, N_{c1}), (M_{c2}, N_{c2}))$ -mutually distinct  $n'$ -paths,  $1 \leq c \leq n-1$ , in  $T$  is  $(\kappa_{ci_{c+1}}, \dots, \kappa_{ci_n})$ , and (i) if  $t$  is in  $\xi_c$ , then  $\kappa_{cit}$  is an identity permutation, and (ii) if  $t$  in  $\{c+1, \dots, n\}$ , is not in  $\xi_c$ , then  $\kappa_{cit} = [2, 1]$ .

### Proof:

We will break up the proof into three parts:

1. Let  $z = g'_1 a_{11}^{i_{111}} b_{11}^{i_{112}} a_{12}^{i_{1r1}} b_{12}^{i_{1r2}} \dots a_{r1}^{i_{1r1}} b_{r1}^{i_{1r2}} a_{r2}^{i_{1r2}} b_{r2}^{i_{1r2}} g_1 \dots g'_{1n} a_{11}^{i_{n'11}} b_{11}^{i_{n'12}} a_{12}^{i_{n'12}} b_{12}^{i_{n'12}} \dots a_{r1}^{i_{n'r1}} b_{r1}^{i_{n'r1}} a_{r2}^{i_{n'r2}} b_{r2}^{i_{n'r2}} g_{1n} g_1 \alpha$ , where  $r = n-1$ , be an  $(n, n')$ -segment sentence in  $L$ , for values of  $i_{111}, i_{112}, \dots, i_{1r1}, i_{1r2}, \dots, i_{n'11}, i_{n'12}, \dots, i_{n'r1}, i_{n'r2}$ , specified as follows:  $i_{s11} = i_{s12} = \dots = i_{sr1} = i_{sr2} = 1$ ,  $1 \leq s \leq n'$ ,  $s \neq j$ , and  $i_{j11} = i_{j12} = \dots = i_{jr1} = i_{jr2} = n'.q$ ,



where  $j$  is the constant referred to in Lemma 2.3.2, and  $q$  is the constant referred to in Theorem 2.1.1. Note that the string  $z$  is now completely specified.

Consider some  $r$  and  $m$ ,  $1 \leq r \leq n-1$ ,  $1 \leq m \leq 2$ , and let  $z_1 = a_{rm}^{n'} \cdot q b_{rm}^{n'} \cdot q$ , where the symbols occurring in  $z_1$  are of type  $(1, j)$ .

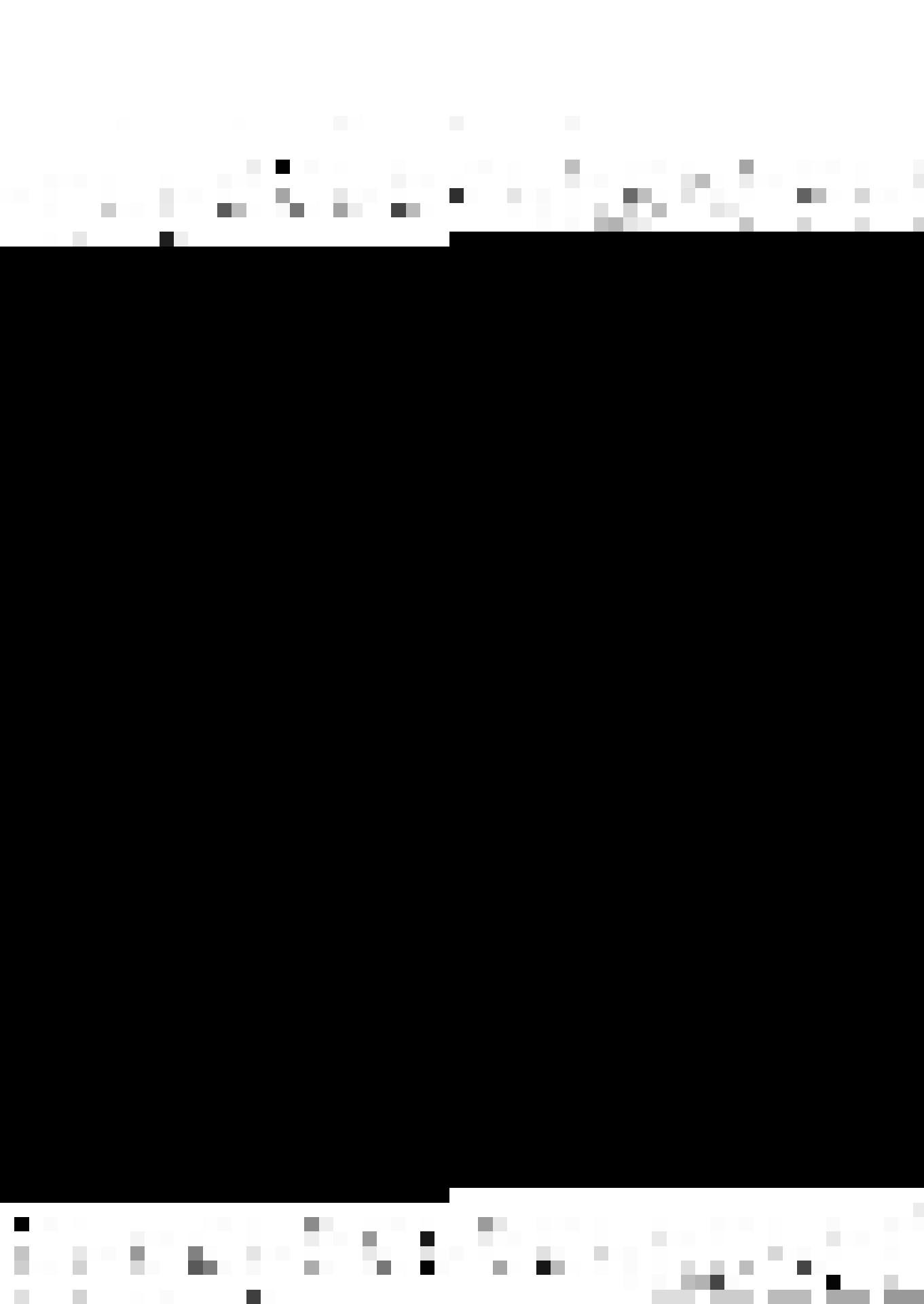
Now, from Theorem 2.1.1, it follows that if the positions occupied by the symbols comprising  $z_1$  are designated 'distinguished', then:

(a) There are two  $n'$ -nodes in  $T$ , say,  $M = \langle M_1, \dots, M_n \rangle$  and  $N = \langle N_1, \dots, N_n \rangle$ , such that (i) there is an  $(M, N)$ -related  $n'$ -path in  $T$ , (ii) for some  $s$ ,  $1 \leq s \leq n'$ ,  $l$ -result( $T_s^{M_s - N_s}$ ) or  $r$ -result( $T_s^{M_s - N_s}$ ) contains a distinguished position, and (iii) for each  $i$ ,  $1 \leq i \leq n'$ , the length of the longest substring of result( $T_i^{M_i}$ ) which contains distinguished positions only, is less than or equal to  $q$ .

(b)  $z$  can be represented as  $z = u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n$ , where (i)  $(v_1 w_1 x_1, \dots, v_n w_n x_n)$ , is the  $n'$ -result of the subtree of  $T$  rooted at  $M$ , and (ii)  $(w_1, \dots, w_n)$ , is the  $n'$ -result of the subtree of  $T$  rooted at  $N$ , such that  $z(i) = u_1 v_1^i w_1 x_1^i y_1 \dots u_n v_n^i w_n x_n^i y_n$ , is also in  $L$ , for all  $i \geq 0$ .

In view of (a) above, (b) may be modified to:

(b')  $z$  can be represented as:  $z = u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n$ , where  $(v_1 w_1 x_1, \dots, v_n w_n x_n)$ , is the  $n'$ -result of the subtree of



$T$  rooted at  $M$ , and (i) for some  $s$ ,  $1 \leq s \leq n'$ ,  $v_s$  or  $x_s$  contains a distinguished position, (ii) for each  $i$ ,  $1 \leq i \leq n'$ , the length of the longest substring of  $v_i w_i x_i$  which contains distinguished positions only is less than or equal to  $q$ , such that  $z(i) = u_1 v_1^i w_1 x_1^i y_1 \dots u_n v_n^i w_n x_n^i y_n$ , is also in  $L$  for all  $i \geq 0$ .

Let the set  $X$  comprise of those  $v_i$ 's and  $x_i$ 's,  $1 \leq i \leq n'$ , which contain, as substrings, nonempty strings comprising of distinguished positions only (i.e. contain a type  $(1,j)$  occurrence of  $a_{rm}$  or a type  $(1,j)$  occurrence of  $b_{rm}$ ).

It is easy to prove the following two statements:  $s_1$ , and  $s_2$  (see Appendix B).

$s_1$ : Any element of  $X$  contains and contains only type  $(1,j)$  occurrence of  $a_{rm}$ , or contains and contains only type  $(1,j)$  occurrence of  $b_{rm}$ .

$s_2$ : A type  $(s',j)$ ,  $1 \leq s' \leq n$ , occurrence of  $a_{rm}$  (respectively,  $b_{rm}$ ) must be in  $V_{s'}$ , where  $V_{s'}$  is  $v_i$  or  $x_i$ ,  $1 \leq i \leq n'$ , such that  $V_{s'}$  contains and contains only type  $(s',j)$  occurrences of  $a_{rm}$  (respectively,  $b_{rm}$ ).

Now, from the statement  $s_2$ , and the fact that the  $j$  we are using is the one referred to in Lemma 2.3.2, we have that for some sequence  $\gamma_{rm}: i_1, \dots, i_n$ ,  $1 \leq i_1 < \dots < i_n \leq n'$ ,



$v_{i_s}$ , contains and contains only type  $(s', j)$  occurrences of  $a_{rm}$ , and  $x_{i_s}$ , contains and contains only type  $(s', j)$  occurrences of  $b_{rm}$ ,  $1 \leq s' \leq n$ .

Again, in view of Lemma 2.3.2, it follows that

$$\psi_{ik} = \psi_{rm}, \text{ for all } i, k, 1 \leq i \leq n-1, 1 \leq k \leq 2.$$

3. Now, in view of Lemma 2.1.3 and Lemma 2.1.4, there are  $((M_{i1}, N_{i1}), (M_{i2}, N_{i2}))$ -mutually distinct  $n'$ -paths,  $1 \leq i \leq n-1$ , in  $T$  such that if the  $n'$ -result of the subtree rooted at  $M_{is}$ ,  $1 \leq s \leq 2$ , is  $(X_1, \dots, X_n)$ , then  $X_{i_s} = v_{i_s}, w_{i_s}, x_{i_s}$ ,  $1 \leq s' \leq n$ , where  $v_{i_s}$ , contains and contains only type  $(s', j)$  occurrences of  $a_{is}$  and  $x_{i_s}$ , contains and contains only type  $(s', j)$  occurrences of  $b_{is}$ . Let the  $(i_c, \dots, i_n)$ -distribution of the  $((M_{c1}, N_{c1}), (M_{c2}, N_{c2}))$ -mutually distinct  $n'$ -paths,  $1 \leq c \leq n-1$ , in  $T$  be  $(\pi_{ci_{c+1}}, \dots, \pi_{ci_n})$ . In view of Lemma 2.3.1(a), we have that if  $t$  is in  $\mathcal{S}_c$ , then  $\pi_{cit}$  is an identity permutation, and if  $t$  in  $\{c+1, \dots, n\}$  is not in  $\mathcal{S}_c$ , then  $\pi_{cit} = [2, 1]$ .

Hence the result.

Q.E.D.

### Structure based on $\mathcal{U}$ -simplicity:

We now prove a lemma which leads to the main results of this section.

#### Lemma 2.3.4:

There is a language in  $\mathcal{C}_{(n, r, k)}^L$ ,  $n \geq 2$ ,  $k \geq 2$ , which is



not in  $\mathcal{C}_{(n', \sigma')}^L$ , where  $n' \geq n$  and  $\mathcal{P}^L(\sigma', \sigma) = 0$ .

Proof:

1. Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ , and  $\sigma' = \xi'_1, \dots, \xi'_{n'-1}$ .

Let  $L$  be the language  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$ .

Assume, for the time being, that  $L$  is in  $\mathcal{C}_{(n', \sigma')}^L$ .

Let it be generated by an  $(n', \sigma')$ -CG.  $G'$  in normal form.

2. According to Lemma 2.3.3, there exists a sentence  $z$  in  $L$  having the following property: Let  $T = \langle T_1, \dots, T_n, \rangle$ , be an  $n'$ -derivation tree of  $z$  in  $G$ . For some  $i_1, \dots, i_n$ ,  $1 \leq i_1 < \dots < i_n \leq n'$ , and some  $n'$ -nodes  $M_{i1}, N_{i1}, M_{i2}, N_{i2}$ ,  $1 \leq i \leq n-1$ , in  $T$ , there are  $((M_{i1}, N_{i1}), (M_{i2}, N_{i2}))$ -mutually distinct  $n'$ -paths,  $1 \leq i \leq n-1$ , in  $T$ , having the property that if the  $(i_c, \dots, i_n)$ -distribution of the  $((M_{c1}, N_{c1}), (M_{c2}, N_{c2}))$ -mutually distinct  $n'$ -paths,  $1 \leq c \leq n-1$ , in  $T$  is  $(\kappa_{ci_{c+1}}, \dots, \kappa_{ci_n})$  and (i) if  $t$  is in  $\xi_c$ , then  $\kappa_{ci_t}$  is an identity permutation, (ii) if  $t$  in  $\{c+1, \dots, n\}$ , is not in  $\xi_c$ , then  $\kappa_{ci_t} = [2, 1]$ .

3.  $\mathcal{P}^L(\sigma', \sigma) = 0$ , implies that for every choice  $\gamma$  of  $j_1, \dots, j_n$ , such that  $1 \leq j_1 < \dots < j_n \leq n'$ , there is atleast one  $t$ ,  $1 \leq t \leq n-1$ , and atleast one  $s$  in  $\{t+1, \dots, n\}$ , the values of  $t$  and  $s$  possibly depending upon  $\gamma$ , such that  $s$  is not in  $\xi_t$  while  $j_s$  is in  $\xi'_{j_t}$ .

Let  $\gamma = i_1, \dots, i_n$ , which is the sequence referred to in 2 above.



Now, from 2 above,  $s$  is not in  $\xi_t$  implies that  $\pi_{tis} = [2, 1]$  = a non-identity permutation. But  $i_s$  is in  $\xi'_t$ , which according to Lemma 2.1.5 implies that  $\pi_{tis}$  is an identity permutation. This is a contradiction.

To complete the proof, it suffices to show that the language  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$  is in  $C_{(n, \sigma, k)}^L$ . This we show in Appendix C.

Q.E.D.

From Lemma 2.2.1 (also using Theorem 2.2.1) and Lemma 2.3.4, we have:

#### Theorem 2.3.1:

$C_{(n, \sigma, k)}^L$  and  $C_{(n', \sigma')}^L$ , are incomparable, where  $n \geq 2$ ,  $k \geq 2$ ,  $n' > n$ , and  $P^L(\sigma', \sigma) = 0$ .

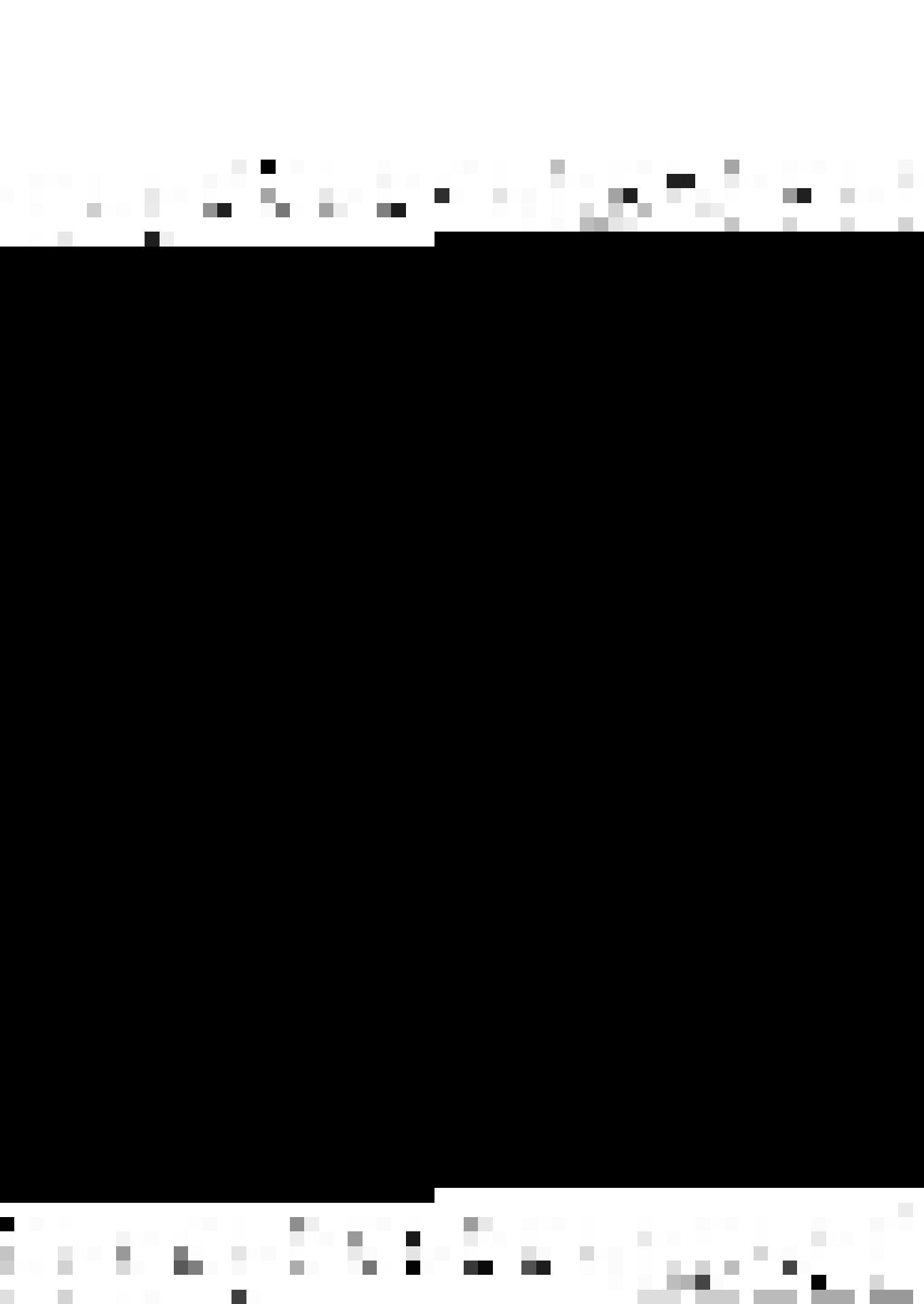
#### Cor. 2.3.1.1:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ . For each  $n'$ , there is a language in  $C_{(n, \sigma, k)}^L$ , which is not in  $\mathcal{S}_n^L$ , where  $\xi_1 \neq \{2, \dots, n\}$ ,  $n \geq 2$ ,  $k \geq 2$ .

#### Cor. 2.3.1.2:

$C_{(m, n, \sigma, k)}^T$  and  $C_{(m', n', \sigma')}^T$ , are incomparable, where  $m' \geq m$ ,  $n' \geq n$ ,  $m'+n' > m+n$ ,  $k \geq 2$ , and one of the following statements is true:

$$(i) P^L(\sigma', \sigma) = 0.$$



(ii)  $m \geq 2$  and  $\rho^L(\sigma'_i, \sigma_i) = 0$ , where  $\sigma_i$  is the  $\mathcal{U}$ -simplicity of the  $m$ -input grammar of some  $(m+n, \sigma, k)$ CG, and  $\sigma'_i$  is the  $\mathcal{U}$ -simplicity of the  $m'$ -input grammar of some  $(m'+n', \sigma')$ CG.

(iii)  $n \geq 2$  and  $\rho^L(\sigma'_0, \sigma_0) = 0$ , where  $\sigma_0$  is the  $\mathcal{U}$ -simplicity of the  $n$ -output grammar of some  $(m+n, \sigma, k)$ CG and  $\sigma'_0$  is the  $\mathcal{U}$ -simplicity of the  $n'$ -output grammar of some  $(m'+n', \sigma')$ CG.

#### Cor. 2.3.1.3:

Let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ . For each  $m', n'$ , there is a translation in  $\mathcal{C}_{(m, n, \sigma, k)}^T$ , which is not in  $\mathcal{B}_{(m', n')}^T$ , where  $\xi_1 \neq \{2, \dots, m+n\}$ , and  $k \geq 2$ .

\*\*\*

Again, from Theorem 2.2.1 and Lemma 2.3.4, we have:

#### Theorem 2.3.2:

$\mathcal{C}_{(n, \sigma, k)}^L \not\subset \mathcal{C}_{(n, \sigma', k')}^L$ , where  $n \geq 2$ ,  $k \geq 2$ ,  $k \geq k'$ ,  $\rho^L(\sigma', \sigma) = 0$ , and  $\rho^L(\sigma, \sigma') = 1$ .

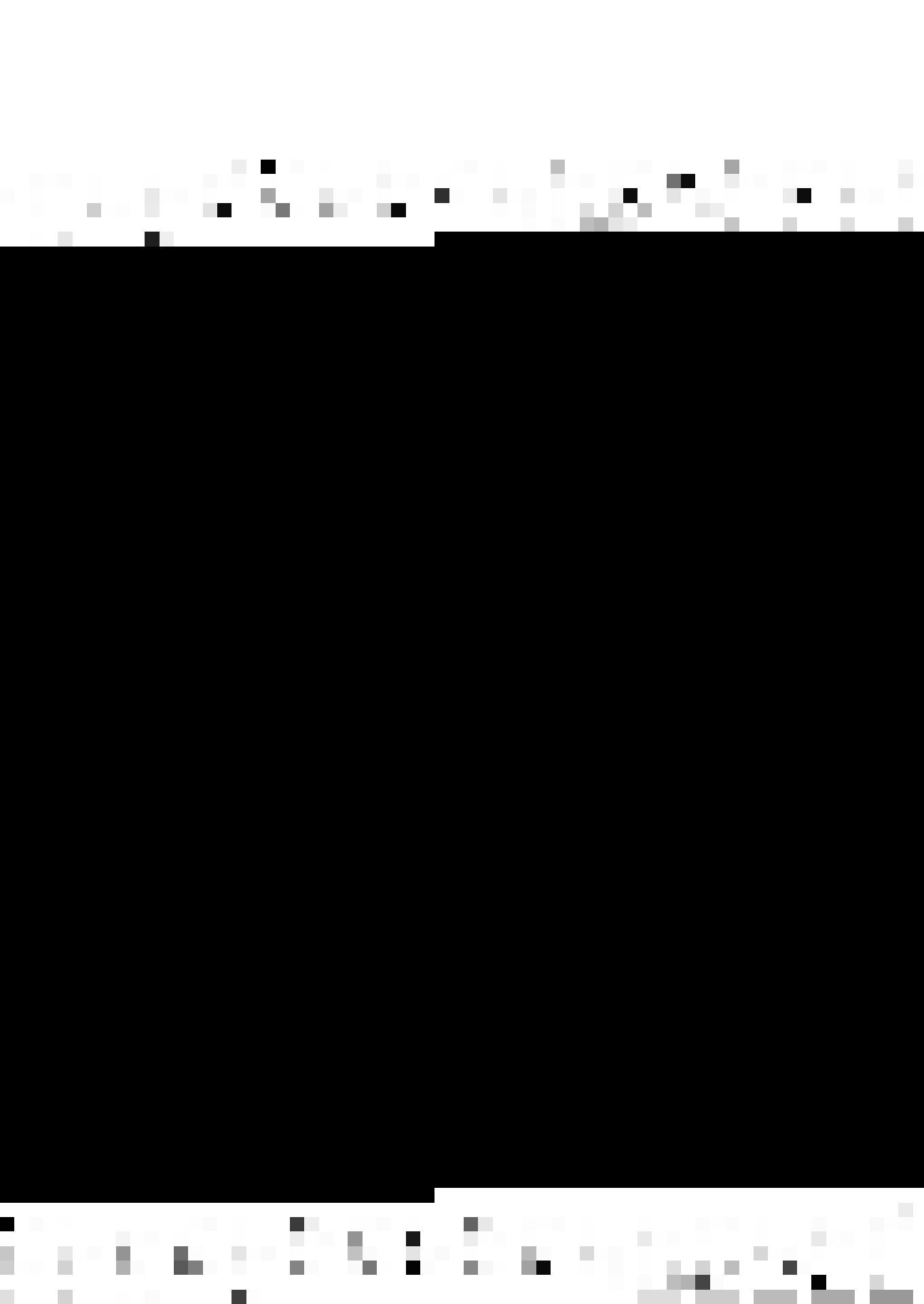
\*\*\*

In view of Lemma 2.1.6, we have:

#### Cor. 2.3.2.1:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ , and  $\sigma' = \xi'_1, \dots, \xi'_{n-1}$ .

$\mathcal{C}_{(n, \sigma, k)}^L \not\subset \mathcal{C}_{(n, \sigma', k')}^L$ , where  $n \geq 2$ ,  $k \geq 2$ ,  $k \geq k'$ , for every  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi_j \subsetneq \xi'_j$ , and for atleast one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi_j \not\subset \xi'_j$ .



Cor 2.3.2.2:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ .

$\mathcal{C}_{(n, \sigma, k)}^L \not\sim \mathcal{B}_{(n, k)}^L$ , where  $\xi_1 \notin \{2, \dots, n\}$ ,  $n \geq 2$ , and  $k \geq 2$ .

Cor. 2.3.2.3:

$\mathcal{C}_{(m, n, \sigma, k)}^T \not\sim \mathcal{C}_{(m, n, \sigma', k')}^T$ , where  $k \geq k'$ ,  $k \geq 2$ ,  $P^L(\sigma', \sigma) = 0$ , and  $P^L(\sigma, \sigma') = 1$ .

Cor. 2.3.2.4:

Let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ .

$\mathcal{C}_{(m, n, \sigma, k)}^T \not\sim \mathcal{B}_{(m, n, k)}^T$ , where  $\xi_1 \notin \{2, \dots, m+n\}$ , and  $k \geq 2$ .

\*\*\*

The last result in this section follows from Lemma 2.3.4.

Theorem 2.3.3:

$\mathcal{C}_{(n, \sigma, k)}^L$  and  $\mathcal{C}_{(n, \sigma', k')}^L$ , are incomparable, where  $k \geq 2$ ,  $k' \geq 2$ ,  $n \geq 2$ ,  $P^L(\sigma', \sigma) = 0$ , and  $P^L(\sigma, \sigma') = 0$ .

Cor. 2.3.3.1:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ , and  $\sigma' = \xi'_1, \dots, \xi'_{n-1}$ .

$\mathcal{C}_{(n, \sigma, k)}^L$  and  $\mathcal{C}_{(n, \sigma', k')}^L$ , are incomparable, where  $n \geq 2$ ,  $k \geq 2$ ,  $k' \geq 2$ , and either of the following two statements is true:



(i) For atleast one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi_j \subsetneq \xi'_j$ , and for atleast one  $j$ ,  $1 \leq j \leq n-1$ ,  $\xi'_j \subsetneq \xi_j$ .

(ii) For atleast one  $j$ ,  $\xi_j$  and  $\xi'_j$  are incomparable.

Cor. 2.3.3.2:

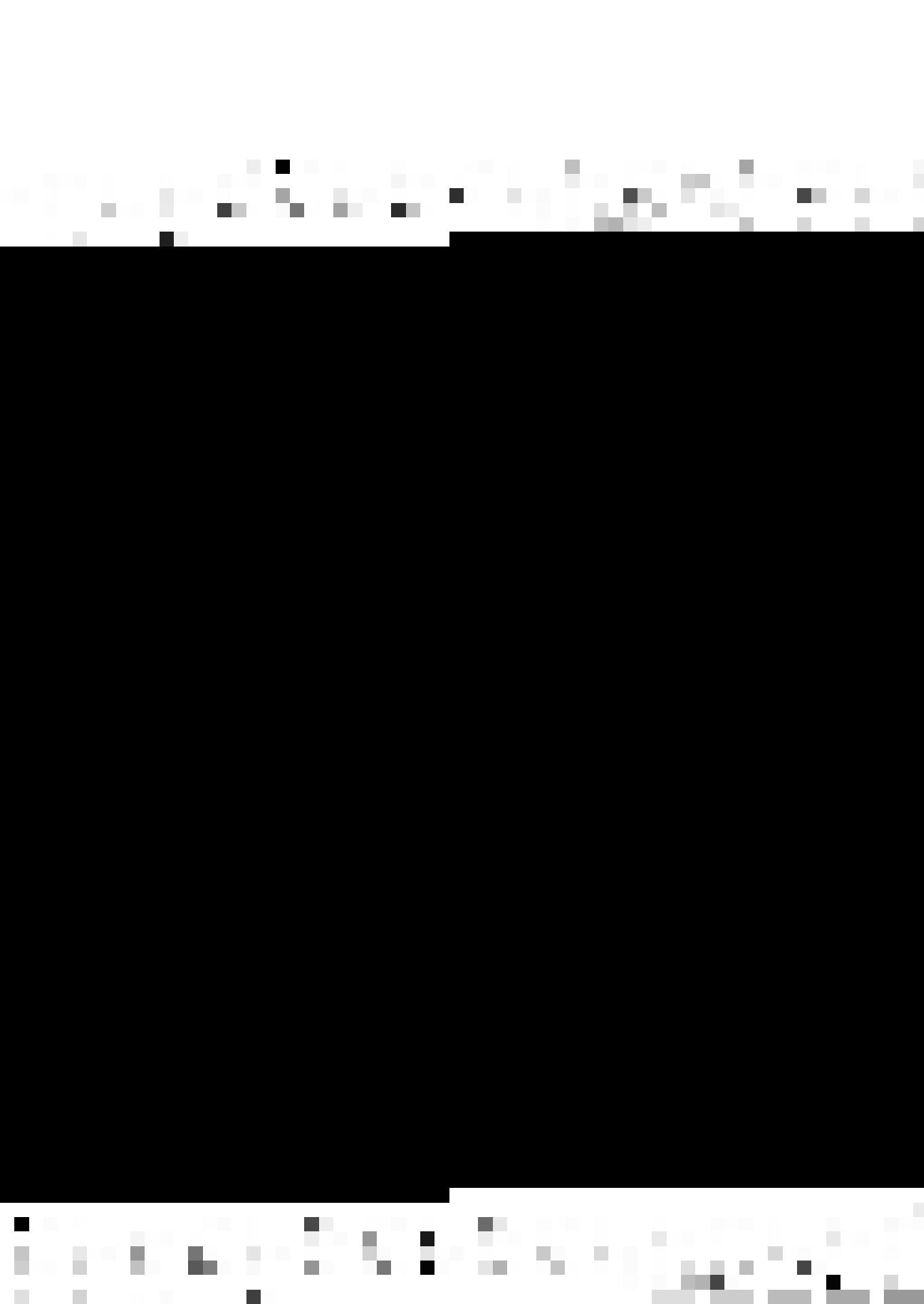
$\mathcal{C}_{(m,n,\sigma,k)}^T$  and  $\mathcal{C}_{(m,n,\sigma',k')}^T$ , are incomparable, where  $k \geq 2$ ,  $k' \geq 2$ ,  $\rho_L(\sigma', \sigma) = 0$ , and  $\rho_L(\sigma, \sigma') = 0$ .

#### 2.4 STRUCTURE RESULTS - III:

In this section, we investigate the structure induced on the coupled languages and coupled translations, by the order of the coupled grammars generating them. We will prove that the order induces an infinite hierarchy on the coupled languages and translations. The methodology adopted is to prove that there is a language in  $\mathcal{C}_{(n,\sigma,k)}^L$ , which is not in  $\mathcal{C}_{(n',k-1)}^L$ ,  $n' \geq n$ ,  $k \geq 4$ ,  $\sigma = \xi_1, \dots, \xi_{n-1}$  is such that  $\xi_1 \neq \{2, \dots, n\}$ . Here also, we need a class of languages which can be used to prove, the above mentioned 'negative' result.

We specify the above mentioned class of languages as follows:

For every  $n \geq 2$ ,  $e \geq 1$ , and  $\xi$  a certain subset of  $\{2, \dots, n\}$ , and  $k \geq 2$ , we define below the language  $L_{(n,\xi,k)}^e$ ; the above mentioned class comprises of all such languages. In order to do so, we shall first define for  $i \geq 1$ ,  $\pi_{2i}$  to be the



permutation  $[i+1, 1, i+2, 2, \dots, 2i, i]$ , and  $\pi_{2i-1}$  to be the permutation:  $[i, 2i-1, 1, 2i-2, \dots, i+1, i-1]$ .

$$\begin{aligned} L_e^e(n, \xi, k) = & \left\{ h_{11}^{i_1} a_1^{j_1} b_1^{l_1} \dots a_k^{i_k} b_k^{j_k} h_{11}^{l_1} \dots h_{tt}^{i_t} a_1^{j_t} b_1^{l_t} \dots \right. \\ & \dots a_k^{i_k} b_k^{j_k} h_{1t}^{l_k} g_1^{h_1} a_{21}^{j_{121}} b_{21}^{l_{121}} \dots \\ & \dots a_{2k}^{j_{121}} b_{2k}^{l_{121}} h_{21}^{l_{21}} \dots h_{2t}^{i_{2t}} a_{21}^{j_{t21}} b_{21}^{l_{t21}} \dots \\ & \dots a_{2k}^{j_{t2k}} b_{2k}^{l_{t2k}} h_{2t}^{l_{2t}} g_2^{h_2} \dots h_{n1}^{i_{n1}} a_{n1}^{j_{1n1}} b_{n1}^{l_{1n1}} \dots \\ & \dots a_{nk}^{j_{1nk}} b_{nk}^{l_{1nk}} h_{n1}^{l_{n1}} \dots h_{nt}^{i_{nt}} a_{n1}^{j_{tn1}} b_{n1}^{l_{tn1}} \dots \\ & \dots a_{nk}^{j_{tnk}} b_{nk}^{l_{tnk}} h_{nt}^{l_{nt}} g_n, \text{ where for all} \end{aligned}$$

$$i, m, r, \quad i \geq 1, \quad 1 \leq m \leq k, \quad 1 \leq r \leq t,$$

(i) For  $r \leq e$ ,  $h_{ir} = g_{ir}$ ,  $h'_{ir} = g'_{ir}$ , and for  $r > e$ , if  $r = e + r'$ ,  $j \geq 1$ ,  $1 \leq r' \leq e$ , then  $h_{ir} = g_{ir'}$ ,  $h'_{ir} = g'_{ir'}$

(ii)  $j_{rsm} = i_{r\hat{\pi}_k(m)}$ ,  $l_{sm} = \hat{\pi}_k(m)$ , for all  $s$  in  $\{2, \dots, n\}$ , such that  $s$  is not in  $\xi$ .

(iii)  $j_{rsm} = i_{rm}$ ,  $l_{sm} = m$ , for all  $s$  in  $\xi$   
 $| i_{rm} \geq 1, t \geq 1 \}^1$

The following example illustrates the definition of the above language.

- Observe the similarities between this language and the language  $L_e^e(n, \xi_1, \dots, \xi_{n-1})$  defined in section 2.3.



Example 2.4.1:

$$\begin{aligned}
 L^2_{(3, \{3\}, 3)} = & \left\{ h'_{11} a_1^{i_{11}} b_1^{i_{11}} a_2^{i_{12}} b_2^{i_{12}} a_3^{i_{13}} b_3^{i_{13}} h_{11} \dots \right. \\
 & \dots h'_{1t} a_1^{i_{t1}} b_1^{i_{t1}} a_2^{i_{t2}} b_2^{i_{t2}} a_3^{i_{t3}} b_3^{i_{t3}} h_{1t} g_1 \\
 & h'_{21} a_3^{i_{13}} b_3^{i_{13}} a_1^{i_{11}} b_1^{i_{11}} a_2^{i_{12}} b_2^{i_{12}} h_{21} \dots \\
 & \dots h'_{2t} a_3^{i_{13}} b_3^{i_{13}} a_1^{i_{11}} b_1^{i_{11}} a_2^{i_{12}} b_2^{i_{12}} h_{2t} g_2 \\
 & h'_{31} a_1^{i_{11}} b_1^{i_{11}} a_2^{i_{12}} b_2^{i_{12}} a_3^{i_{13}} b_3^{i_{13}} h_{31} \dots \\
 & \dots h'_{3t} a_1^{i_{11}} b_1^{i_{11}} a_2^{i_{12}} b_2^{i_{12}} a_3^{i_{13}} b_3^{i_{13}} h_{3t} g_3, \text{ where}
 \end{aligned}$$

for  $r \leq 2$ ,  $h_{ir} = g_{ir}$ ,  $h'_{ir} = g'_{ir}$ , and for  $r > 2$ , if  $r = 2 \cdot j + r'$ ,  
 $j \geq 1$ ,  $1 \leq r' \leq 2$ , then  $h_{ir} = g_{ir'}$ ,  $h'_{ir} = g'_{ir'}$ ,  $1 \leq i \leq 3$

$$| i_{u1}, i_{u2}, i_{u3} \geq 1, i \leq u \leq t, t \geq 1 \} . \quad ***$$

The proof of the following lemma is exactly on similar lines as that of Lemma 2.3.3 and is omitted.

Lemma 2.4.1:

Let  $L$  be the language  $L_{(n, \{k\}, k)}^{n'}$ , for some  $n, \{k\}, k, n, n' \geq n$ . Let it be generated by an  $n'$ -CG  $G$  in normal form. There exists a sentence  $z$  in  $L$  having the following property: Let  $T := \langle T_1, \dots, T_n \rangle$  be an  $n'$ -derivation tree of  $z$  in  $G$ . For some  $i_1, \dots, i_n$ ,  $1 \leq i_1 < \dots < i_n \leq n'$ , there are  $((M_1, N_1), \dots, (M_k, N_k))$ -mutually distinct  $n'$ -paths in  $T$ , having the property that if their  $(i_1, \dots, i_n)$ -distribution is  $(\pi_{i_1}, \dots, \pi_{i_n})$ , and if  $t$  in  $\{2, \dots, n\}$  is not in  $\{i_1, \dots, i_n\}$ , then  $\pi_{i_t} = \pi_k$ . \*\*\*



The following definitions help in simplifying certain statements:

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an  $n$ -derivation tree. If  $T'$  is an  $n$ -derivation tree which is got from  $T$  by relabeling all the nonpendant nodes in each  $T_i$ ,  $1 \leq i \leq n$ , such that if  $M$  is an  $n$ -node in  $T$ , then it is an  $n$ -node in  $T'$  also, and no two  $n$ -nodes in  $T'$  have got the same label, then  $T'$  is said to be got from  $T$  by distinctly relabeling  $T$ .

\*\*\*

Let  $T$  be an  $n$ -derivation tree in an  $(n, \sigma, k)$  CG  $G = (V_N, V_T, P, (S)^n)$ . It is clear that from an inspection of  $T$ , we can say as to which productions have necessarily to be in  $P$ , if at all  $T$  is to be an  $n$ -derivation tree. Call the set of such productions as the 'production set associated with  $T$ '. Now, we have the following definition:

Let  $T = \langle T_1, \dots, T_n \rangle$ , be an  $n$ -derivation tree. Let  $A$  be the label of its  $n$ -root,  $X$  be the set of labels of its  $n$ -nodes and  $Y$  be the set of labels of the pendant nodes of  $T_1, \dots, T_n$ , and  $P$  be the production set associated with  $T$ . Then,  $G = (X, Y, P, (A)^n)$ , is called  $n$ CG associated with  $T$ .

We now prove a lemma which leads to the results of this section.

#### Lemma 2.4.2:

There is a language in  $\mathcal{C}_{(n, \sigma, k)}^L$ , which is not in  $\mathcal{C}_{(n', k-1)}^L$ , where  $n' \geq n$ ,  $k \geq 4$ , and  $\sigma = S_1, \dots, S_{n-1}$ , is such



that  $\xi_1 \neq \{2, \dots, n\}$ .

Proof:

Let  $L$  be the language  $L_{(n, \xi_1, k)}^{n'}$ .

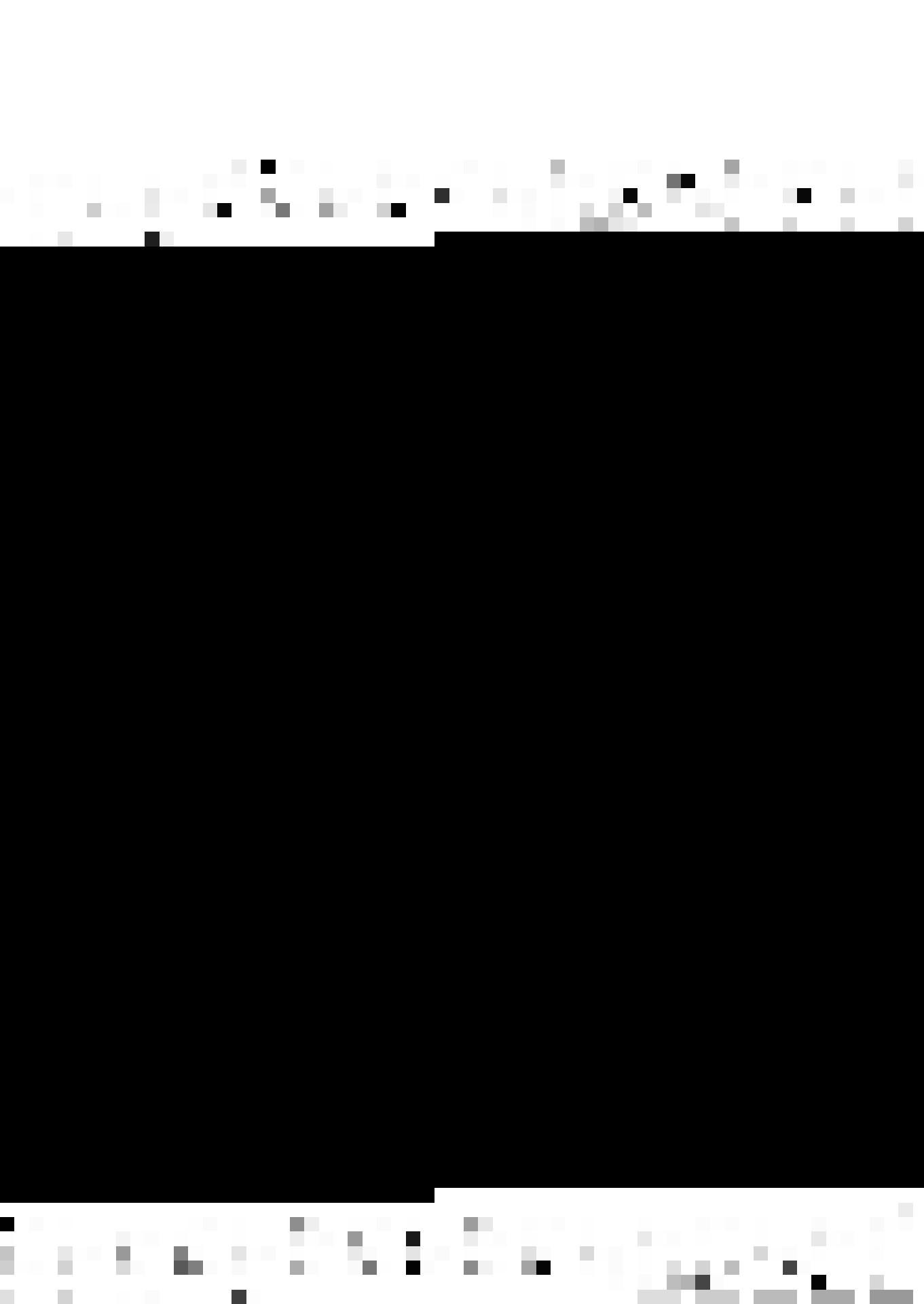
Suppose that  $L$  is in  $\mathcal{C}_{(n', \sigma_r, k-1)}^L$ . Let this language be generated by some  $(n', \sigma_r, k-1)CG G$ , in normal form. Let  $\sigma_r = \xi_1^r, \dots, \xi_{n_r-1}^r$ .

We know that  $\xi_1 \neq \{2, \dots, n\}$ . Let  $s$  in  $\{2, \dots, n\}$ , be not in  $\xi_1$ . Let us now use Lemma 2.4.1. We have that there exists a sentence  $z$  in  $L$  having the following property: Let  $T = \langle T_1, \dots, T_{n'} \rangle$ , be an  $n'$ -derivation tree of  $z$  in  $G$ . For some  $i_1, \dots, i_n$ ,  $1 \leq i_1 < \dots < i_n \leq n'$ , there are  $((M_1, N_1), \dots, (M_k, N_k))$  - mutually distinct  $n'$ -paths in  $T$ , having the property that if their  $(i_1, \dots, i_n)$ -distribution is  $(\pi_{i_2}, \dots, \pi_{i_n})$ , then  $\pi_{i_s} = \pi_k$ .

Let  $\pi$  be the  $(1, i_1)$ -distribution of the above mentioned mutually distinct  $n'$ -paths in  $T$ .

Now, through a sequence of constructions, we will get from  $G$ , 'Syntax directed translation grammar of order  $k-1$ ' (3), 'defining' the translation:

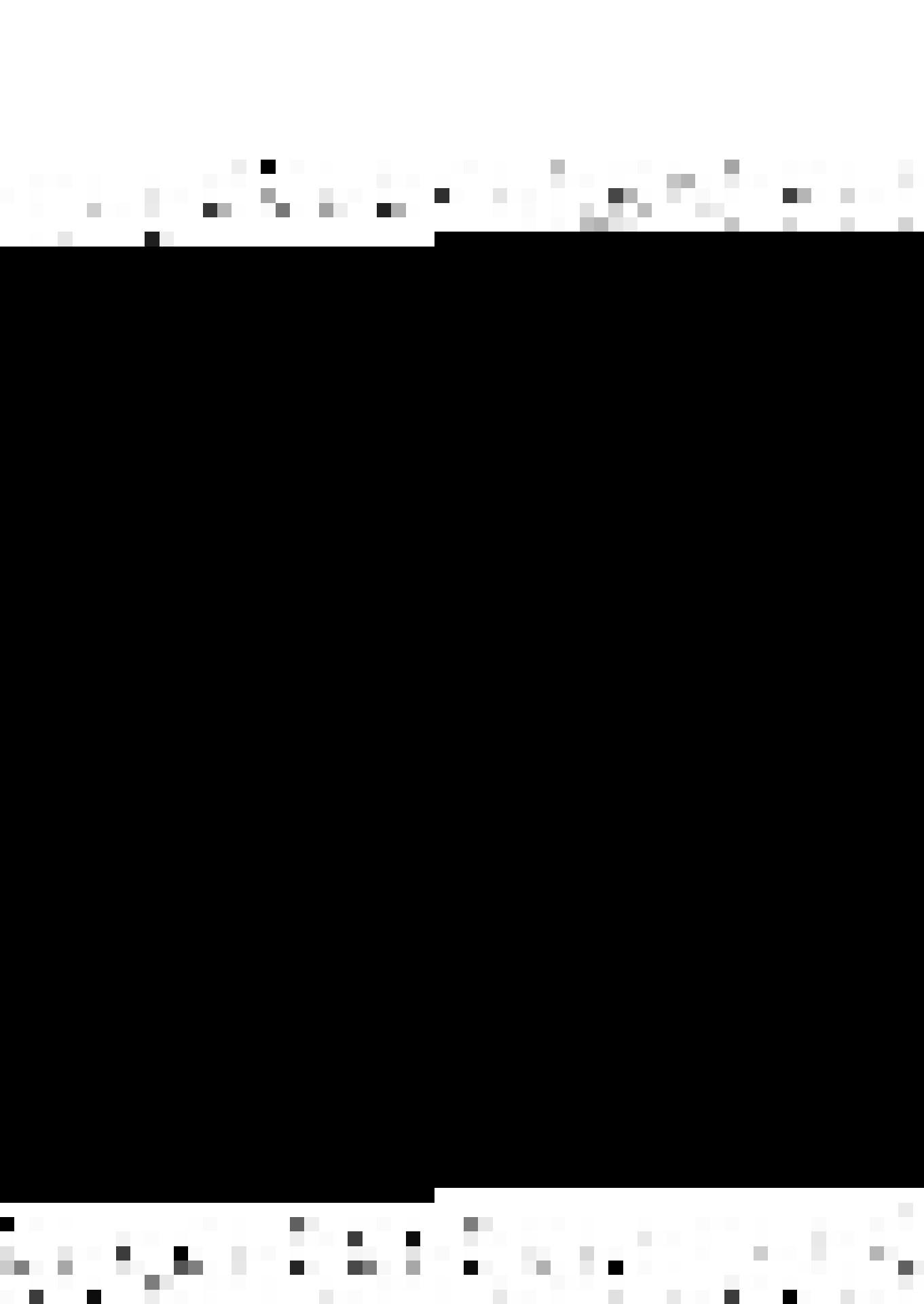
$$T = \left\{ (a_1^{i_1} a_2^{i_2} \dots a_k^{i_k}, b_1^{j_1} b_2^{j_2} \dots b_k^{j_k}) \mid \text{for } 1 \leq m \leq k, i_m \geq 1, b_{\pi_k(m)} = a_m, \text{ and } j_{\pi_k(m)} = i_m \right\} :$$



1. Get an  $n'$ -derivation tree  $T' = \langle T'_1, \dots, T'_{n'} \rangle$  from  $T$  by distinctly relabeling  $T$ .
2. Let  $T'' = \langle T'_{i_1}, T'_{i_s} \rangle$ . Then,  $T''$  is a 2-derivation tree. Corresponding to each  $n'$ -node in  $T'$ , there will be a 2-node in  $T''$ . In particular, we will have the 2-nodes:  $M'_i = \langle M_{ii_1}, M_{ii_s} \rangle$ , and  $N'_i = \langle N_{ii_1}, N_{ii_s} \rangle$ ,  $1 \leq i \leq k$ , where  $M_i = \langle M_{i1}, \dots, M_{in'} \rangle$  and  $N_i = \langle N_{i1}, \dots, N_{in'} \rangle$ ,  $1 \leq i \leq k$ , are  $n'$ -nodes in  $T'$ , referred to above.
3. Replace the labels of all pendant nodes in  $T'_{i_1}, T'_{i_s}$ , by  $\epsilon$ .
4. If for  $i$ ,  $1 \leq i \leq k$ ,  $A_i$  is the label of  $M'_i$ , then replace the subtree in  $T''$  rooted at  $M'_i$  by the 2-derivation tree shown in Figure 2.4.1. Let  $T_f = \langle T_{f_1}, T_{f_2} \rangle$ , be the resulting 2-derivation tree.
5. Get the grammar associated with  $T_f$ . Let it be,  
 $G_f = (V_N, V_T, P, (S, S))$ .
6. Reduce  $G_f$  to  $G'_f$ , where<sup>1</sup>  $T_1(G_f) = T_1(G'_f)$ , and all non-terminating productions in  $G'_f$  are of the type:  $A \rightarrow (\alpha_1, (\alpha_2, \bar{\alpha}_2))$ ; using Lemma 1.2.3, if there is some nonterminating production in  $G_f$  which is not of this type. Clearly  $G'_f$  is a 'Syntax directed translation grammar of order  $k-1$ ' (3) and it defines the translation  $T$ .

---

1.  $T_1(G_f)$  is the 1-translation generated by  $G_f$ . Similarly,  $T_1(G'_f)$  is the 1-translation generated by  $G'_f$ . (See Section 1.2).



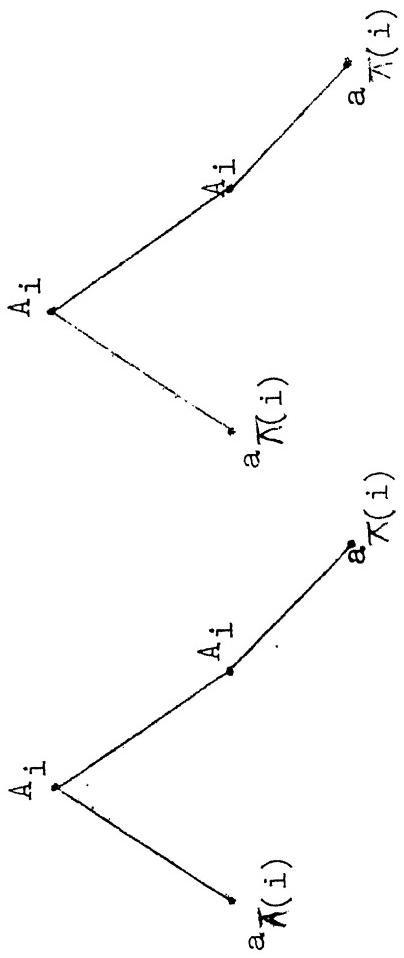
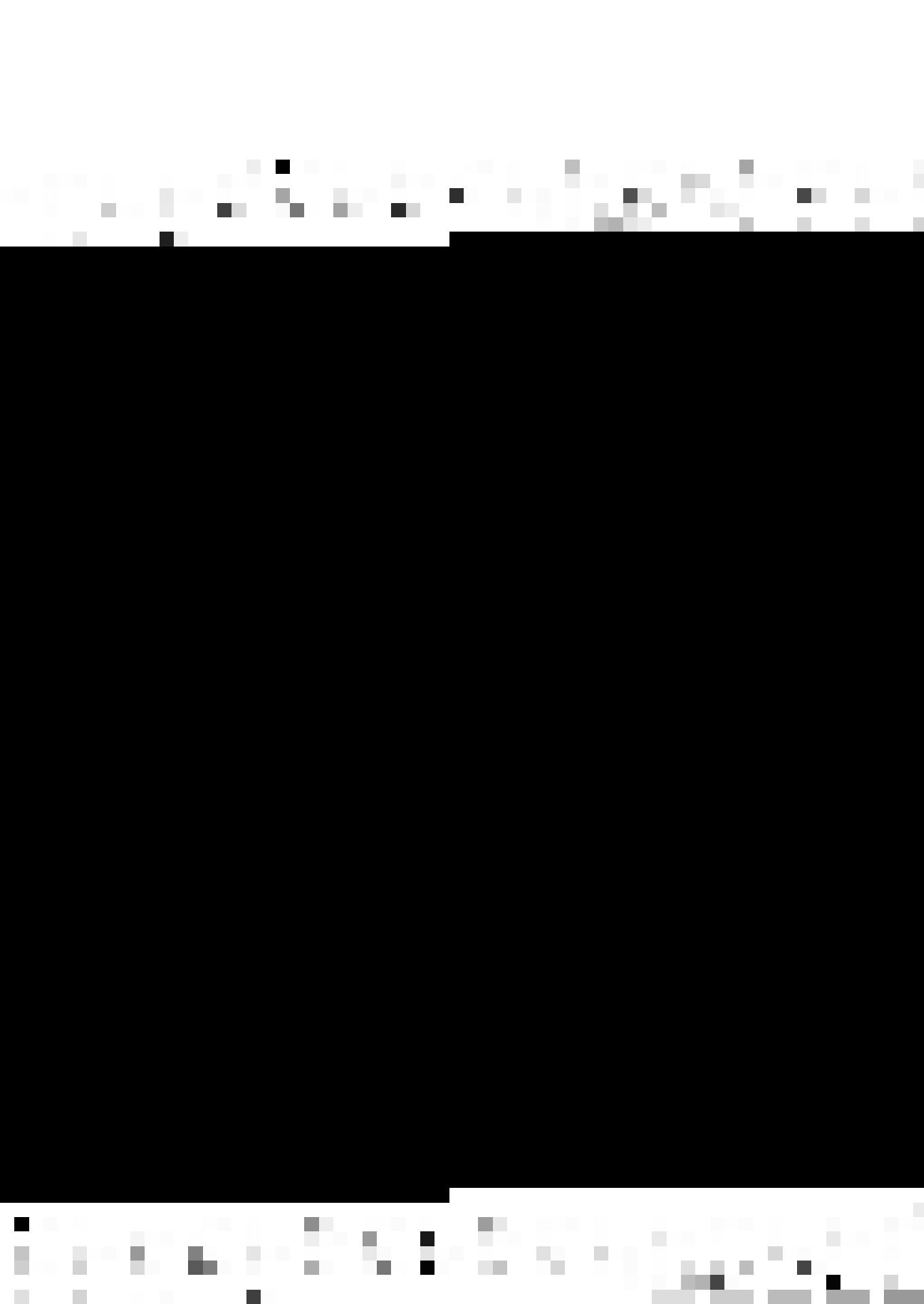


Figure 2.4.1



The existence of  $G'_f$  defining the translation  $T$  is a contradiction to the proof of a theorem (Theorem 3.3) in (3).

To complete the proof, it suffices to show that  $L$  is in  $\mathcal{C}_{(n, \sigma, k)}^L$ . The construction of an  $(n, \sigma, k)$  CG which generates the required language is given in Appendix D.

Hence the result.

Q.E.D.

The following result follows from Lemma 2.2.1, Theorem 2.2.1, and Lemma 2.4.2:

Theorem 2.4.1:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ .

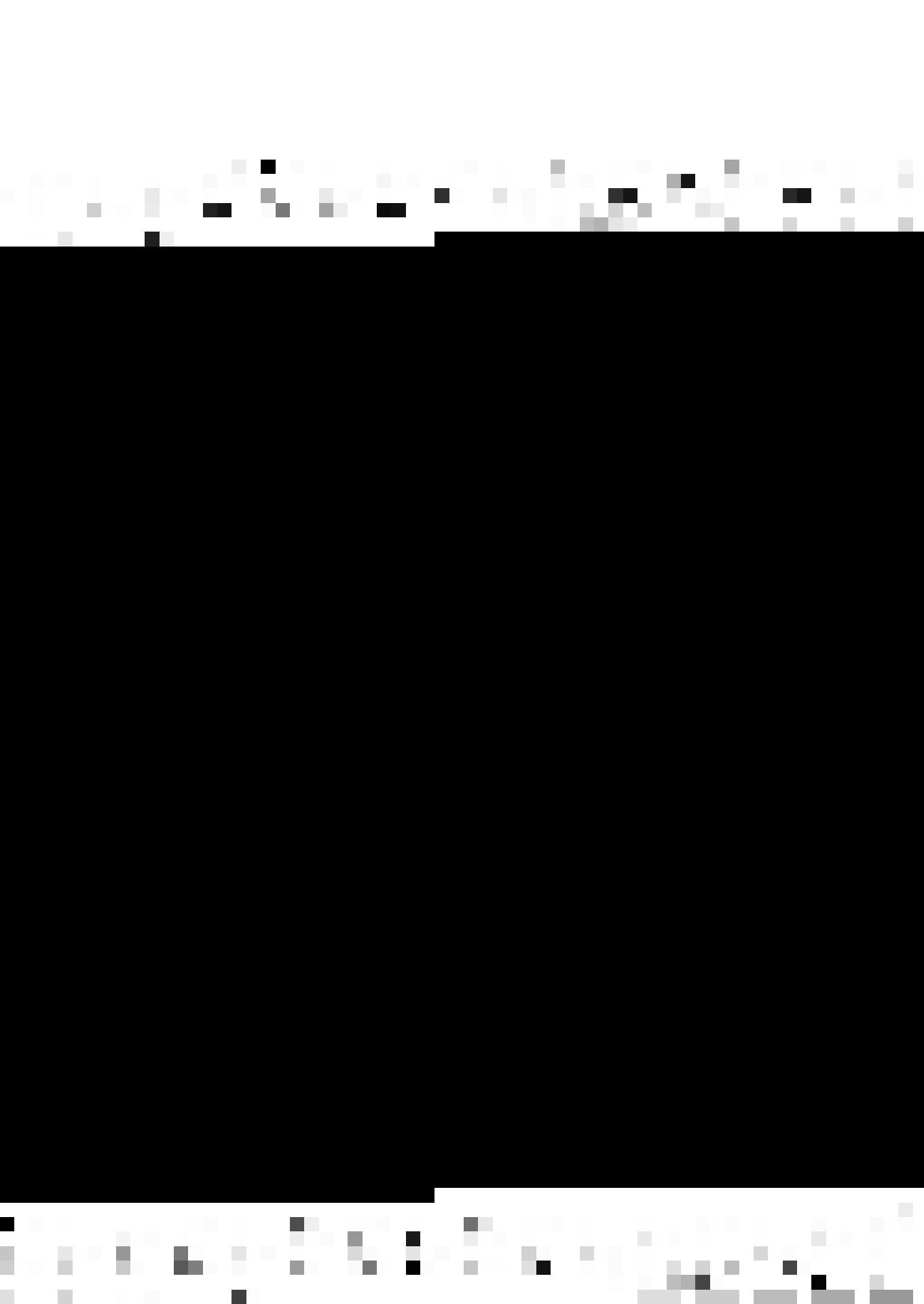
- (a)  $\mathcal{C}_{(n, \sigma, k')}^L \not\sim \mathcal{C}_{(n, \sigma, k)}^L$ , where  $n \geq 2$ ,  $\xi_1 \neq \{2, \dots, n\}$ ,  $k' \geq 4$ , and  $k' > k$ .
- (b)  $\mathcal{C}_{(n, \sigma, k')}^L$  and  $\mathcal{C}_{(n', k)}^L$ , are incomparable, where  $n' > n$ ,  $\xi_1 \neq \{2, \dots, n\}$ ,  $n \geq 2$ ,  $k' \geq 4$ , and  $k' > k$ .

Cor. 2.4.1.1:

Let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ .

- (a)  $\mathcal{C}_{(m, n, \sigma, k')}^T \not\sim \mathcal{C}_{(m, n, \sigma, k)}^T$ , where  $\xi_1 \neq \{2, \dots, m+n\}$ ,  $k' \geq 4$ ,  $k' > k$ .
- (b)  $\mathcal{C}_{(m, n, \sigma, k')}^T$  and  $\mathcal{C}_{(m', n', k)}^T$ , are incomparable, where  $m' \geq m$ ,  $n' \geq n$ ,  $m'+n' > m+n$ ,  $k' > k$ ,  $k' \geq 4$ , and  $\xi_1 \neq \{2, \dots, m+n\}$

\*\*\*



From Theorem 2.2.1, Lemma 2.3.4, and Lemma 2.4.2, we have the following result:

Theorem 2.4.2:

Let  $\sigma = \xi_1, \dots, \xi_{n-1}$ .

$C_{(n, \sigma', k')}^L$  and  $C_{(n, \sigma, k)}^L$ , are incomparable, where  $n \geq 2$ ,  $\xi_1 \notin \{2, \dots, n\}$ ,  $P^L(\sigma', \sigma) = 0$ ,  $k' > k$ , and  $k' \geq 4$ .

\*\*\*

In view of Theorem 1.2.1, we have:

Cor. 2.4.2.1:

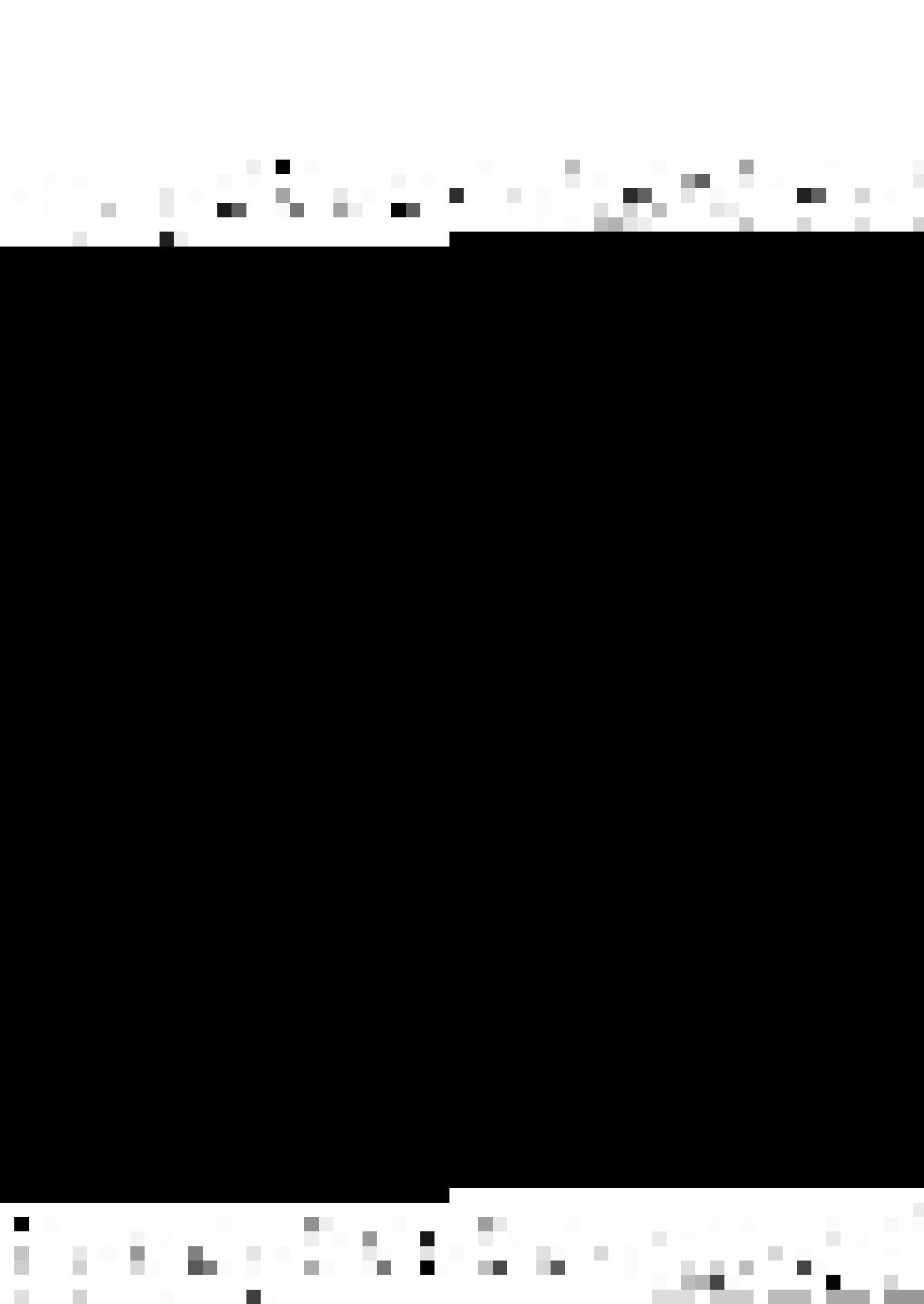
Let  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ .

$C_{(m, n, \sigma', k')}^T$  and  $C_{(m, n, \sigma, k)}^T$ , are incomparable, where  $\xi_1 \notin \{2, \dots, m+n\}$ ,  $k' \geq 4$ ,  $k' > k$ , and  $P^L(\sigma', \sigma) = 0$ .

\*\*\*

Observe that in the above corollary,  $P^L(\sigma', \sigma) = 0$ , implies that  $P^T(\sigma', \sigma, m, m) = 0$ .

Finally, we come to the following set of results:



Theorem 2.4.3:

- (a)  $\mathcal{S}_{(n,k)}^L = \mathcal{S}_{(n,k-1)}^L$ ,  $k > 3$ ;
- (b)  $\mathcal{S}_{(m,n,k)}^T = \mathcal{S}_{(m,n,k-1)}^T$ ,  $k \geq 3$ ;
- (c)  $\mathcal{S}_{(1,1,2)}^T \supset \mathcal{S}_{(1,1,1)}^T$ ;
- (d)  $\mathcal{C}_{(1,1,2)}^T \supset \mathcal{C}_{(1,1,1)}^T$ ;
- (e)  $\mathcal{C}_{(1,1,3)}^T = \mathcal{C}_{(1,1,2)}^T$ ;
- (f)  $\mathcal{C}_{(2,3)}^L = \mathcal{C}_{(2,2)}^L$ .

\*\*\*

In the above theorem, (a) and (b) follow from Theorem 1.4.2 and Cor. 2.2.1.1. (c) and (d) follow from the fact that the class of context-free languages properly contains the class of linear languages (9). (e) and (f) follow from the fact that a 'syntax directed translation of order  $k$ ' is the same as  $\mathcal{C}_{(1,1,k)}^T$ , and the corresponding result in (3).

2.5 CONCLUSIONS:

In this chapter, we have investigated the structure induced by the degree,  $\mu$ -simplicity and order of coupled grammars on the languages and translations generated by them and their interrelationships (these results are given as theorems in sections 2.2 through 2.4). Actually we have proved the following structure results and the various other structure results follow from these:



(a) There is an  $(n+1, \sigma, k)$ -coupled language (or an  $(n+1, k)$ -simple coupled language),  $k \geq 1$ , which is not an  $n$ -coupled language (Lemma 2.2.1 and Lemma 2.2.2).

(b) Any  $(n, \sigma, k)$ -coupled language is also an  $(n', \sigma', k')$ -coupled language, where  $n' \geq n$ ,  $k' \geq k$ , and  $\rho^L(\sigma', \sigma) = 1$  (Theorem 2.2.1).

(c) There is an  $(n, \sigma, k)$ -coupled language,  $n \geq 2$ ,  $k \geq 2$ , which is not an  $(n', \sigma')$ -coupled language, where  $n' \geq n$  and  $\rho^L(\sigma', \sigma) = 0$  (Lemma 2.3.4).

(d) There is an  $(n, \sigma, k)$ -coupled language which is not an  $(n', k-1)$ -coupled language, where  $n' \geq n$ ,  $k \geq 4$ , and  $\sigma = \xi_1, \dots, \xi_{n-1}$ , is such that  $\xi_1 \notin \{2, \dots, n\}$  (Lemma 2.4.2).

In view of the fact that 'syntax directed translation' (4) is the same as  $(1, 1)$ -coupled translation and 'simple syntax directed translation' is the same as  $(1, 1)$ -simple coupled translation, it follows from a corollary of one of the results of this chapter (Cor. 2.3.2.4) that the class of syntax directed translations properly contains the class of simple syntax directed translations. Thus we have provided an alternative proof of this result to that given in (4).

We have proved a pumping theorem for  $(n, \sigma, k)$ -coupled languages (Theorem 2.1.1). This result is not limited in its use to proving structure results only. In fact, we use this result in Chapter 4 also, for proving certain other results.



From the pumping theorem for  $(n, \sigma, k)$ -coupled languages, a more powerful result (Cor. 2.1.1.2) for context-free languages than the uvwxy theorem follows, in that it allows decomposition of a sentence relative to any given substring  $z_1$ . This result is not, however, as general as the Ogden's lemma (27).

In this chapter, we have used extensively the concepts of  $n$ -derivation trees and related concepts, introduced in Chapter 1. We have found these concepts useful for proving various results. We have introduced in this chapter the concepts of language simplicity predicate and translation simplicity predicate which play an important role in proving various structure results.

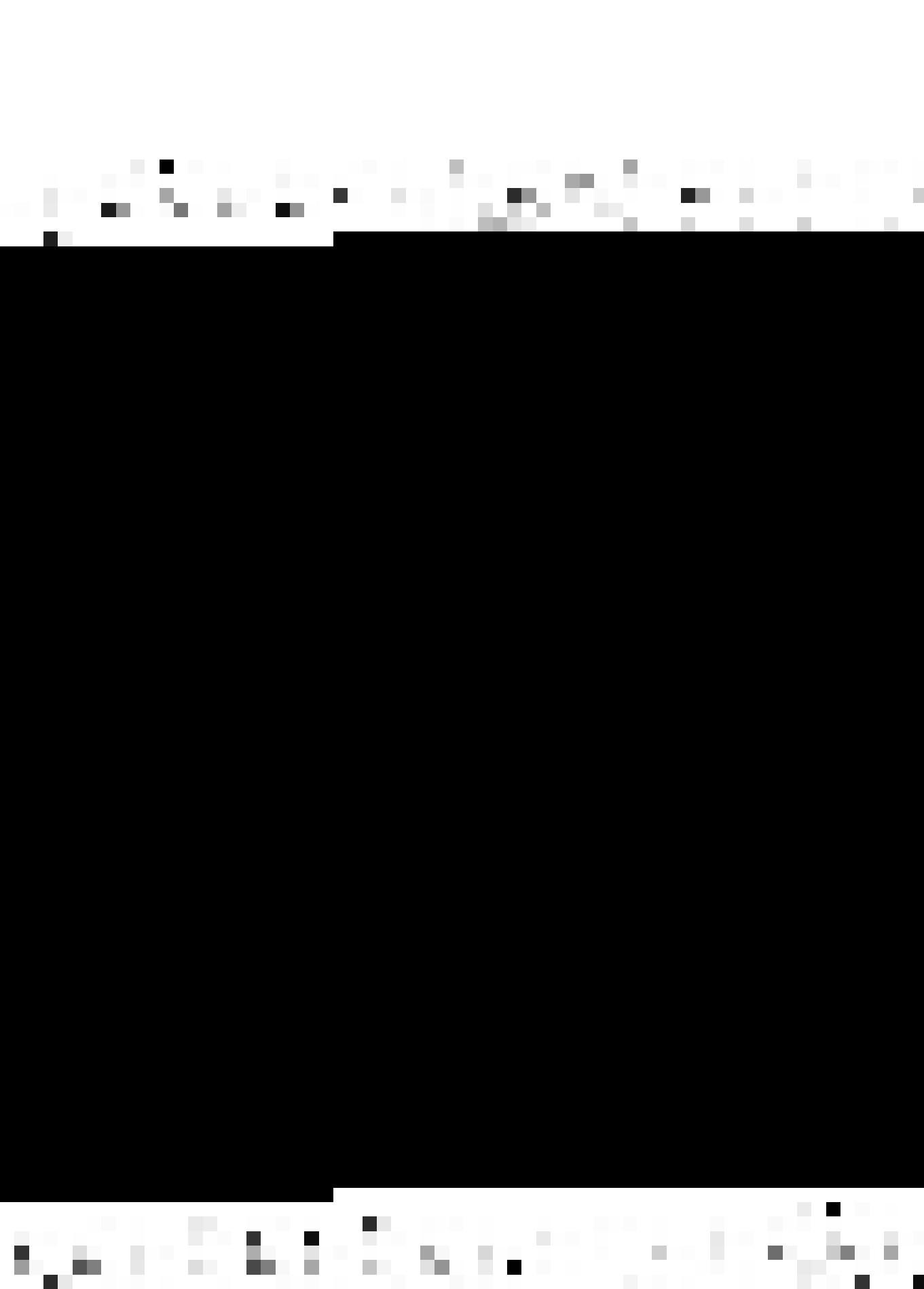
We feel that the following conjectures are true:

(1)  $\mathcal{C}_{(n, \sigma, 3)}^L \not\sim \mathcal{C}_{(n, \sigma, 2)}^L$ , and  $\mathcal{C}_{(m, n-m, \sigma, 3)}^T \not\sim \mathcal{C}_{(m, n-m, \sigma, 2)}^T$ , where  $n \geq 3$  and  $\sigma = \xi_1, \dots, \xi_{n-1}$  is such that  $\xi_1 \neq \{2, \dots, n\}$ ,

(2)  $\mathcal{S}_{(n, 2)}^L \not\sim \mathcal{S}_{(n, 1)}^L$ , and  $\mathcal{S}_{(m, n-m, 2)}^T \not\sim \mathcal{S}_{(m, n-m, 1)}^T$ , where  $n \geq 2$ ;

(3)  $\mathcal{C}_{(m, n, \sigma, k)}^T$  and  $\mathcal{C}_{(m', n', \sigma')}^T$ , are incomparable, where  $m' \geq m$ ,  $n' \geq n$ ,  $m'+n' > m+n$ ,  $k \geq 2$ , and  $\rho^T(\sigma', \sigma, m', m) = 0$ ;

(4)  $\mathcal{C}_{(m, n, \sigma', k')}^T$  and  $\mathcal{C}_{(m, n, \sigma, k)}^T$ , are incomparable, where  $k' \geq 4$ ,  $k' > k$ ,  $\rho^T(\sigma', \sigma, m, m) = 0$ , and  $\sigma = \xi_1, \dots, \xi_{m+n-1}$ , is such that  $\xi_1 \neq \{2, \dots, m+n\}$ ; and



(5) C 28.

We believe that (1) and (2) can be proved through the use of the concept of n-derivation trees, and the conjectures (3), (4) and (5) can be proved by extending the scheme of proof that we have developed for proving Lemma 2.3.4.

The degree,  $\mu$ -simplicity and order are some of the complexity measures of coupled grammars. These measures can also be indicators of the 'difficulty' of writing coupled grammars which model certain classes of languages or translations, or of the 'complexity' of systems based on these grammars. The results obtained in this chapter throw light on how restrictions on coupled grammars in terms of these complexity measures affect their generative power. The general implication of these results is that such restriction(s) on the value(s) of one or more of these complexity measures of coupled grammars may entail loss in their generative power and such restrictions on the values of different complexity measures may entail different losses in their generative power.



## CHAPTER 3

### MACHINE CHARACTERIZATIONS

In this chapter, we discuss machine models which characterize the class of  $n$ -simple coupled languages and the class of  $(1,n)$ -coupled translations.

In section 3.1, we propose and discuss a class of machines which we call  $n$ -turn generalized checking automata, and prove some simplification results of these machines. In section 3.2, we prove the equivalence between the class of languages accepted by  $n$ -turn generalized checking automata and the class of  $n$ -simple coupled languages. In section 3.3, we discuss  $n$ -pushdown assemblers, which we propose as machine characterizations of  $(1,n)$ -coupled translations. Finally, we sum up our conclusions in section 3.4.

#### 3.1 $n$ -TURN GENERALIZED CHECKING AUTOMATA (39):

The definition of the  $n$ -turn generalized checking automaton, as we propose below, is a generalization of the definition, of  $n$ -turn checking automaton (35). One can also look at this definition as the minimal extension of the features of 'restricted' pushdown automata — which cannot overprint on or erase a symbol on the pushdown stack, so that they can serve as machine models of  $n$ -simple coupled languages. If we put the constraint that the  $n$ -turn generalized checking automata must



simulate the leftmost derivations of n-simple coupled grammars, we get some pointers as to why these machines are defined the way we have defined them. (Also note in this connection that the class of languages accepted by 1-turn generalized checking automata has to be the same as the class of context-free languages).

### Basic Definitions:

We shall first discuss informally the features of an n-turn generalized checking automaton and how these features can be used by the machine i.e. the 'working' of the machine. Later we shall give its formal definition.

Let  $n \geq 1$ . An n-turn generalized checking automaton (nGCA) (see Figure 3.1.1) is a nondeterministic machine which consists of:

a finite control;

an input tape and a 1-way input head, the input tape being 'read-only';

a counter which can contain numbers from 0 to  $n$  and which can get incremented by 1 at a time (cannot get decremented); and

a stack and a stack head which can move both ways but in a restricted manner: The stack is infinite in one direction only, say to the right, and is divided into cells. The blank cells are said to contain the 'blank' symbols. There are two



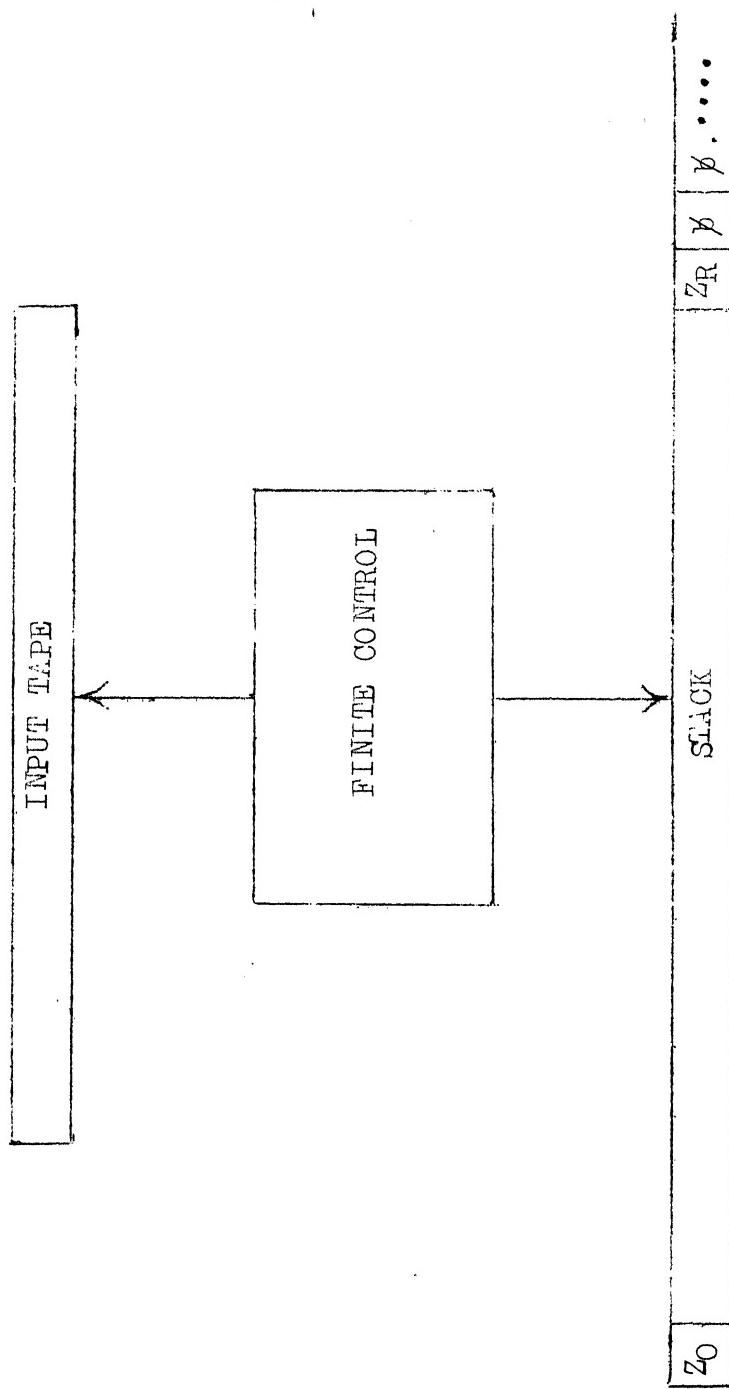


Figure 3.1.1.



special stack symbols, one of which acts as the 'left end marker' (marking the left end of the stack) and the other acts as the 'stack string right end marker' (marking at any instant the right end of the nonblank portion of the stack).

The nGCA can be either in the 'checking' (read-only) mode (which will be the case when the counter contains a number different from zero) or in the 'adding' (read/print)<sup>1</sup> mode (which will be the case when the counter contains zero).

The input head does not move when the machine is in the adding mode, or when the machine is in the checking mode and the stack head is reading the stack string right end marker. In other situations, the input head may either choose not to move or may move right by one cell.

The stack head cannot erase a stack symbol or overprint on a stack symbol. It can, however, overprint on a blank symbol. It can read any symbol in the nonblank portion of the stack. In the adding mode, the stack head can print  $\epsilon^2$  (respectively, can print a string of stack symbols - possibly  $\epsilon$ , which does not contain the left end marker and the stack string right end marker), when reading the stack string right end marker (respectively, when reading a symbol other than the stack string right end marker, say Z). This it does as follows: The machine

1. We will use the terms: print, store, add, to mean the same thing.
2. Printing  $\epsilon$  means not printing anything.



shifts all nonblank symbols to the right of  $Z$  to create just enough blank cells immediately to the right of the cell occupied by  $Z$  so as to accommodate the string to be printed and prints the string in these cells, in order from the left. Note that the stack contents will always be  $Z_0$  followed on its right by a nonblank string of stack symbols (which does not contain the left end marker and the stack string right end marker) followed by the stack string right end marker followed by an infinite sequence of blank symbols.

As far as motion of the stack head is concerned, the stack head can choose one of the two alternatives:

1. Move right by one cell (in which case the machine does not increment the counter).

2. Move to the leftmost cell of the stack--irrespective of the current position of the stack head (in which case, the machine increments the counter).

Formally, an nGCA is a 10-tuple:

$M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, Z_0, Z_R, F)$ , where,

1.  $Q$  is a finite set of states, with  $q_0$  as the start state;

2.  $\Sigma$  is a finite set of input symbols;

3.  $\Gamma$  is a finite set of nonblank stack symbols which includes the left end marker and the stack string right end marker,  $Z_0$  and  $Z_R$ , respectively;



4. C is a counter capable of containing any integer i,  
 $0 \leq i \leq n$ ;

5. F, the set of final (halting) states is a subset of Q;

6.  $\delta_a$  is either a mapping from  $(Q \times (\Gamma - \{z_R\}))$  to finite subsets of  $(Q \times (\Gamma - \{z_0, z_R\})^*) \times \{0, 1\}$  or a mapping from  $(Q \times \{z_R\})$  to finite subsets of  $(Q \times \{\epsilon\} \times \{0, 1\})$ ; and

7.  $\delta_c$  is either a mapping from  $(Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma - \{z_R\}) \times (C - \{0\}))$  to finite subsets of  $(Q \times \{0, 1\})$  or a mapping from  $(Q \times \{\epsilon\} \times \{z_R\} \times (C - \{0\}))$  to finite subsets of  $(Q \times \{0, 1\})$ .

\*\*\*

$\delta_a$  is the 'next move mapping' for the condition when the counter contains zero. Under this condition, the input head does not move, any string of stack symbols (possibly  $\epsilon$ ) may be printed on the stack (without destroying any information), the machine may change state, the stack head may move right by one cell (in which case the counter is not incremented) or the stack head may move to the leftmost cell of the stack (in which case, the counter is incremented by 1).

If  $\delta_a(q, Z)$  contains  $(q', z_1 \dots z_k, 0)$  (respectively,  $(q', z_1 \dots z_k, 1)$ ),  $k \geq 0^1$ , and the counter contains zero, then if the symbol scanned by the stack head is  $Z$  and M is in state  $q$ , the machine may go into state  $q'$ , shift rightwards

---

1.  $k = 0$  means that  $z_1 \dots z_k = \epsilon$ .



all stack symbols which are to the right of  $Z$  (if any) by  $k$  cells, print the string  $Z_1 \dots Z_k$  in order from left in the cells (one symbol in one cell) immediately to the right of the one occupied by  $Z$ , move the stack head right by one cell (respectively, move the stack head to the leftmost cell of the stack and increment the counter), without moving the input head.

$\delta_c$  is the 'next move mapping' for the condition when the counter contains a number different from zero. Under this condition, the input head may move by one cell or may not move, the machine may change state, the stack head may move right by one cell (in which case the counter is not incremented) or may move to the leftmost cell of the stack (in which case the counter is incremented by 1).

If  $\delta_c(q, a, Z, j)$  contains  $(q', 0)$  (respectively,  $(q', 1)$ ),  $1 \leq j \leq n$  (respectively,  $1 \leq j \leq n-1$ ), the counter contains  $j$ , the symbol scanned by the stack head is  $Z$ , the symbol scanned by the input head is  $a$  (in case  $a = \epsilon$ , then it does not matter what symbol the input head is scanning), and the machine is in state  $q$ , then the machine may go into state  $q'$ , move the input head by one cell - in case  $a \neq \epsilon$ , or not move it - in case  $a = \epsilon$ , move the stack head right by one cell (respectively, move the stack head to the leftmost cell of the stack and increment the counter).



We call the move of the nGCA when it is in the adding mode as an adding move and its move when it is in the checking mode as a checking move. Note that a checking move cannot precede an adding move.

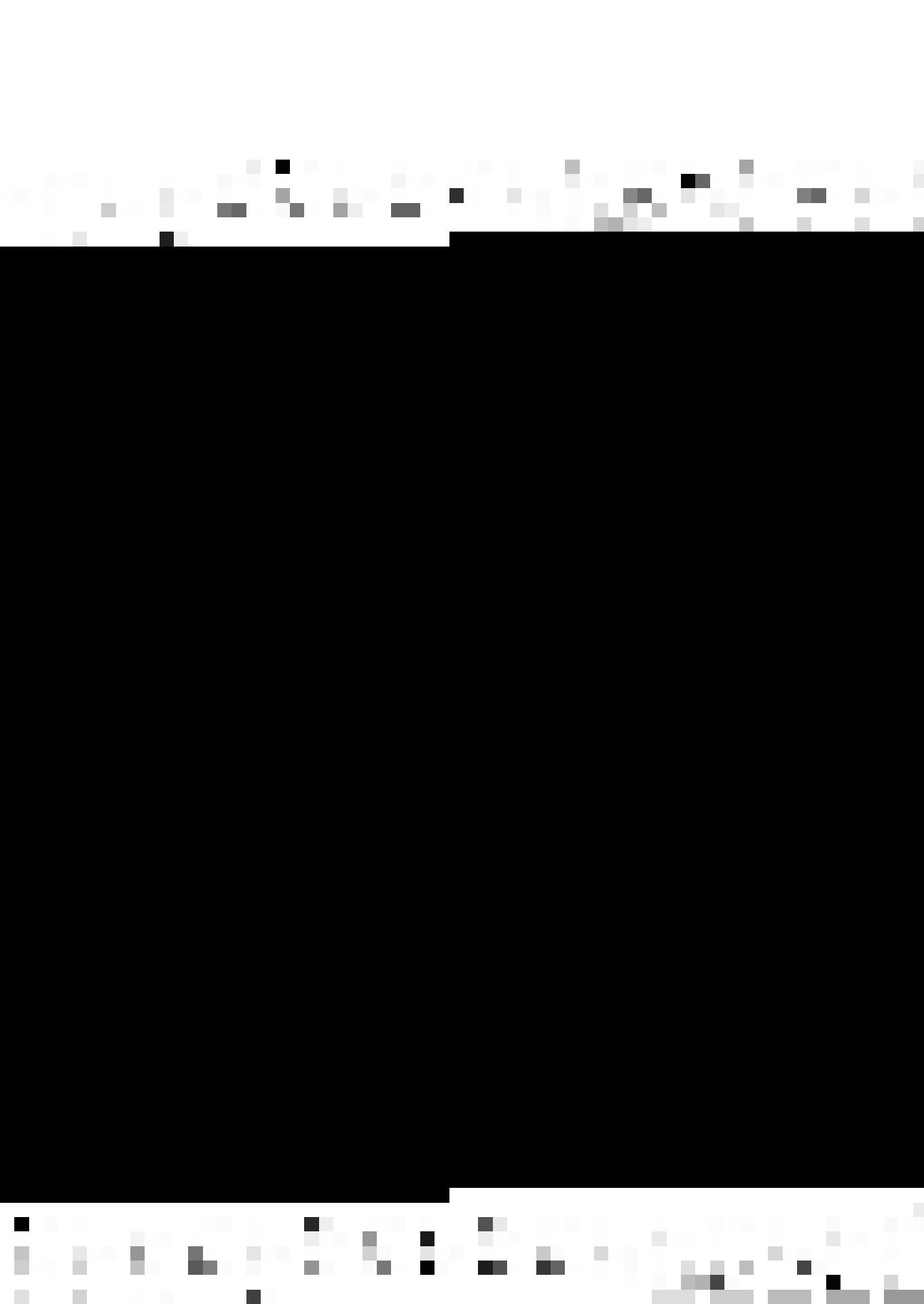
We define a 'configuration' of an nGCA  $M = (Q, \Sigma^*, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$ , with input in  $\Sigma^*$ , to be a combination of the state of finite control, contents of the portion of the stack including and to the left of the cell scanned by the stack head if the stack head is scanning a blank symbol, otherwise the contents of the nonblank portion of the stack (the position of the stack head being shown in both cases by the special symbol ' $\uparrow$ '; the cell scanned by the stack head is the one immediately to the left of this symbol), the contents of the nonblank portion of the input tape including and to the right of the cell being scanned by the input head, and content of the counter.

Formally, a configuration of  $M$  is a 4-tuple:  $(q, w, \alpha, j)$ , where  $q$  is in  $Q$ ,  $w$  is in  $\Sigma^*$ ,  $\alpha$  is either in  $z_0(\Gamma - \{z_0, z_R\})^* \uparrow (\Gamma - \{z_0, z_R\})^* z_R$ , or in  $z_0(\Gamma - \{z_0, z_R\})^* z_R \uparrow^1$ .

We say that:

$$(q_1, w_1, \alpha_1, j_1) \vdash_M^a (q_2, w_2, \alpha_2, j_2),^2$$

1. Note that the machine cannot proceed any further from this configuration. This is why the configuration as we have defined above need not and does not describe situations in which the stack head is reading a blank symbol which is not the leftmost blank symbol in the stack.
2. ' $M$ ' may be dropped from under ' $\vdash$ ' whenever, because of the context no ambiguity can result.



if configuration  $(q_1, w_1, \alpha_1, j_1)$  can become  $(q_2, w_2, \alpha_2, j_2)$  by a single adding move. Also, if  $(q_1, w_1, \alpha_1, j_1)$  can become  $(q_2, w_2, \alpha_2, j_2)$  by some number of adding moves (including zero moves), then we write:

$$(q_1, w_1, \alpha_1, j_1) \xrightarrow{a^*} (q_2, w_2, \alpha_2, j_2).$$

If  $(q_1, w_1, \alpha_1, j_1)$  can become  $(q_2, w_2, \alpha_2, j_2)$  by a single checking move, we say:

$$(q_1, w_1, \alpha_1, j_1) \xrightarrow{c} (q_2, w_2, \alpha_2, j_2).$$

Similarly, if by some number of checking moves (including zero moves),  $(q_1, w_1, \alpha_1, j_1)$  can become  $(q_2, w_2, \alpha_2, j_2)$ , we write:

$$(q_1, w_1, \alpha_1, j_1) \xrightarrow{c^*} (q_2, w_2, \alpha_2, j_2).$$

M accepts an input w if it can be written as  $w_1 \dots w_n$ ,  $w_i$ ,  $1 \leq i \leq n$ , in  $\Sigma^*$ , and the initial configuration:  $(q_0, w_1 \dots w_n, z_0 \uparrow z_R, 0)$ , is such that

$$(q_0, w_1 \dots w_n, z_0 \uparrow z_R, 0)$$

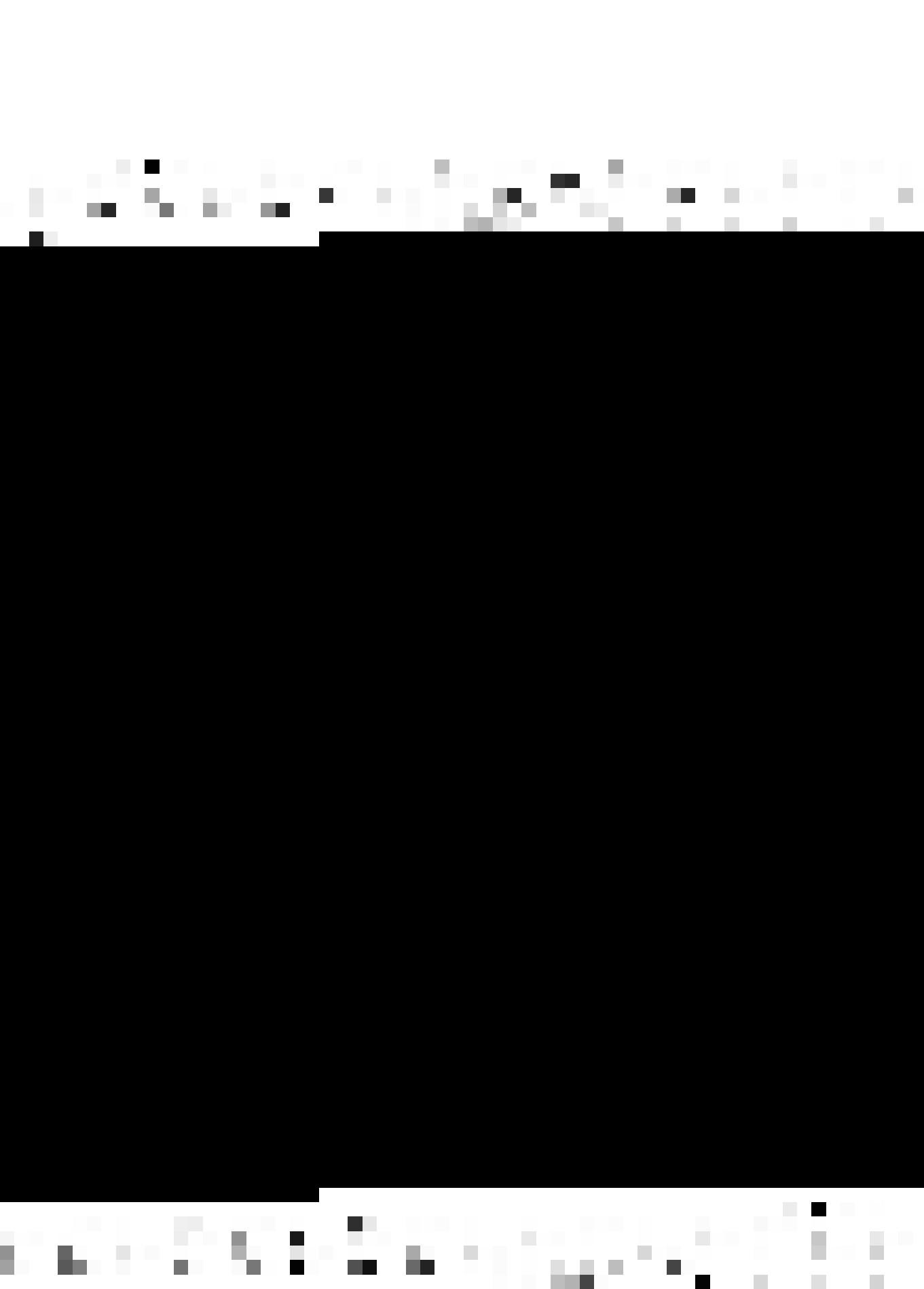
$$\xrightarrow{a^*} (q_1, w_1 \dots w_n, z_0 \uparrow \alpha \alpha' z_R, 1)$$

$$\xrightarrow{c^*} (q_2, w_2 \dots w_n, z_0 \uparrow \alpha \alpha' z_R, 2)$$

$$\xrightarrow{c^*} (q_n, w_n, z_0 \uparrow \alpha \alpha' z_R, n)$$

$$\xrightarrow{c^*} (q, \epsilon, z_0 \uparrow \alpha' z_R, n),$$

for any  $\alpha, \alpha', q_1, \dots, q_n$ , and some  $q$  in  $F$ .



4.  $\delta_c(q_1, \epsilon, z_0, 1) = \{(q_1, 0)\}$ ,  
 (Skip  $z_0$ ).
5.  $\delta_c(q_1, a, z_1, 1) = \{(q_1, 0)\}$ ,  
 (Check a against  $z_1$  when the counter contains 1).
6.  $\delta_c(q_1, b, z_2, 1) = \{(q_1, 0)\}$ ,  
 (Check b against  $z_2$  when the counter contains 1).
7.  $\delta_c(q_1, \epsilon, z_R, 1) = \{(q_1, 1)\}$ ,  
 (Sense the right end of the nonblank portion of the stack and increment the counter).
8.  $\delta_c(q_1, \epsilon, z_0, 2) = \{(q_1, 0)\}$ ,  
 (Skip  $z_0$ ).
9.  $\delta_c(q_1, c, z_1, 2) = \{(q_1, 0)\}$ , and  
 (Check c against  $z_1$  when the counter contains 2).
10.  $\delta_c(q_1, \epsilon, z_2, 2) = \{(q_f, 0)\}$ .  
 (Sense the end of the  $z_1$  symbols on the stack and enter the final halting state).

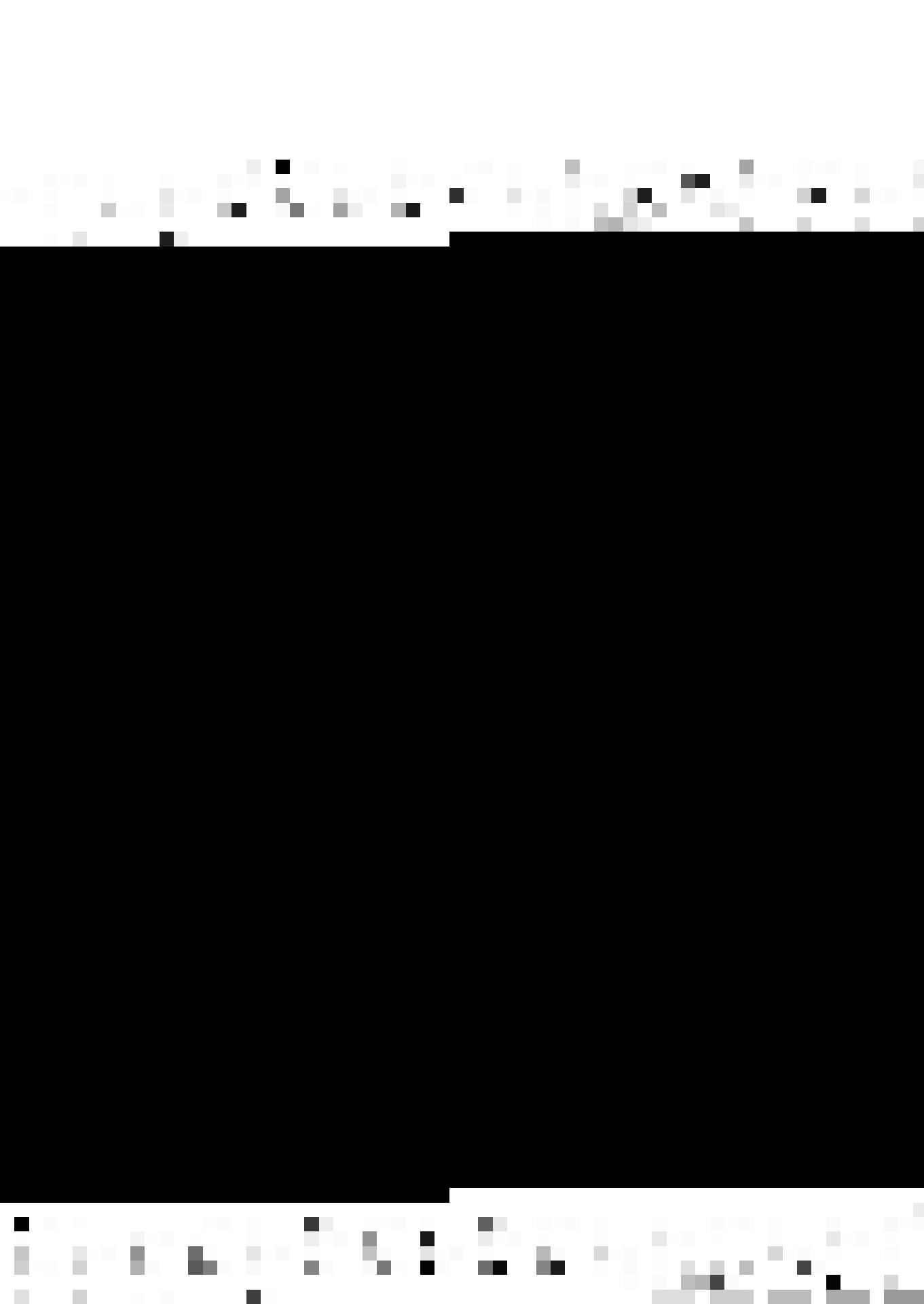
M accepts the input if all symbols get checked properly, machine enters the final halting state and the input is exhausted.

Let us see how the sentence aabbcc is accepted by M by tracing the configurations of the machine.

$(q_0, aabbcc, z_0 \uparrow z_R, 0)$

$\xrightarrow{a} (q_0, aabbcc, z_0 z_1 \uparrow z_2 z_R, 0)$

(by rule 1)



$\overleftarrow{H}^a(q_0, aabbcc, z_0z_1z_1 \uparrow z_2z_2z_R, 0)$ 

(by rule 2)

 $\overleftarrow{H}^a(q_1, aabbcc, z_0 \uparrow z_1z_1z_2z_2z_R, 1)$ 

(by rule 3)

 $\overleftarrow{H}^c(q_1, aabbcc, z_0z_1 \uparrow z_1z_2z_2z_R, 1)$ 

(by rule 4)

 $\overleftarrow{H}^c(q_1, abbcc, z_0z_1z_1 \uparrow z_2z_2z_R, 1)$ 

(by rule 5)

 $\overleftarrow{H}^c(q_1, bbcc, z_0z_1z_1z_2 \uparrow z_2z_R, 1)$ 

(by rule 5)

 $\overleftarrow{H}^c(q_1, bcc, z_0z_1z_1z_2z_2 \uparrow z_R, 1)$ 

(by rule 6)

 $\overleftarrow{H}^c(q_1, cc, z_0z_1z_1z_2z_2z_R \uparrow, 1)$ 

(by rule 6)

 $\overleftarrow{H}^c(q_1, cc, z_0 \uparrow z_1z_1z_2z_2z_R, 2)$ 

(by rule 7)

 $\overleftarrow{H}^c(q_1, cc, z_0z_1 \uparrow z_1z_2z_2z_R, 2)$ 

(by rule 8)



$$\vdash^c (q_1, c, z_0 z_1 z_1 \uparrow z_2 z_2 z_R, 2)$$

(by rule 9)

$$\vdash^c (q_1, \epsilon, z_0 z_1 z_1 z_2 \uparrow z_2 z_R, 2)$$

(by rule 9)

$$\vdash^c (q_f, \epsilon, z_0 z_1 z_1 z_2 z_2 \uparrow z_R, 2)$$

(by rule 10)

### Simplification Results:

The following lemma simplifies some of the later proofs. It relates the number of symbols stored by an nGCA to the number of moves it takes the machine to 'cross over' all of these symbols, in the adding mode of the machine.

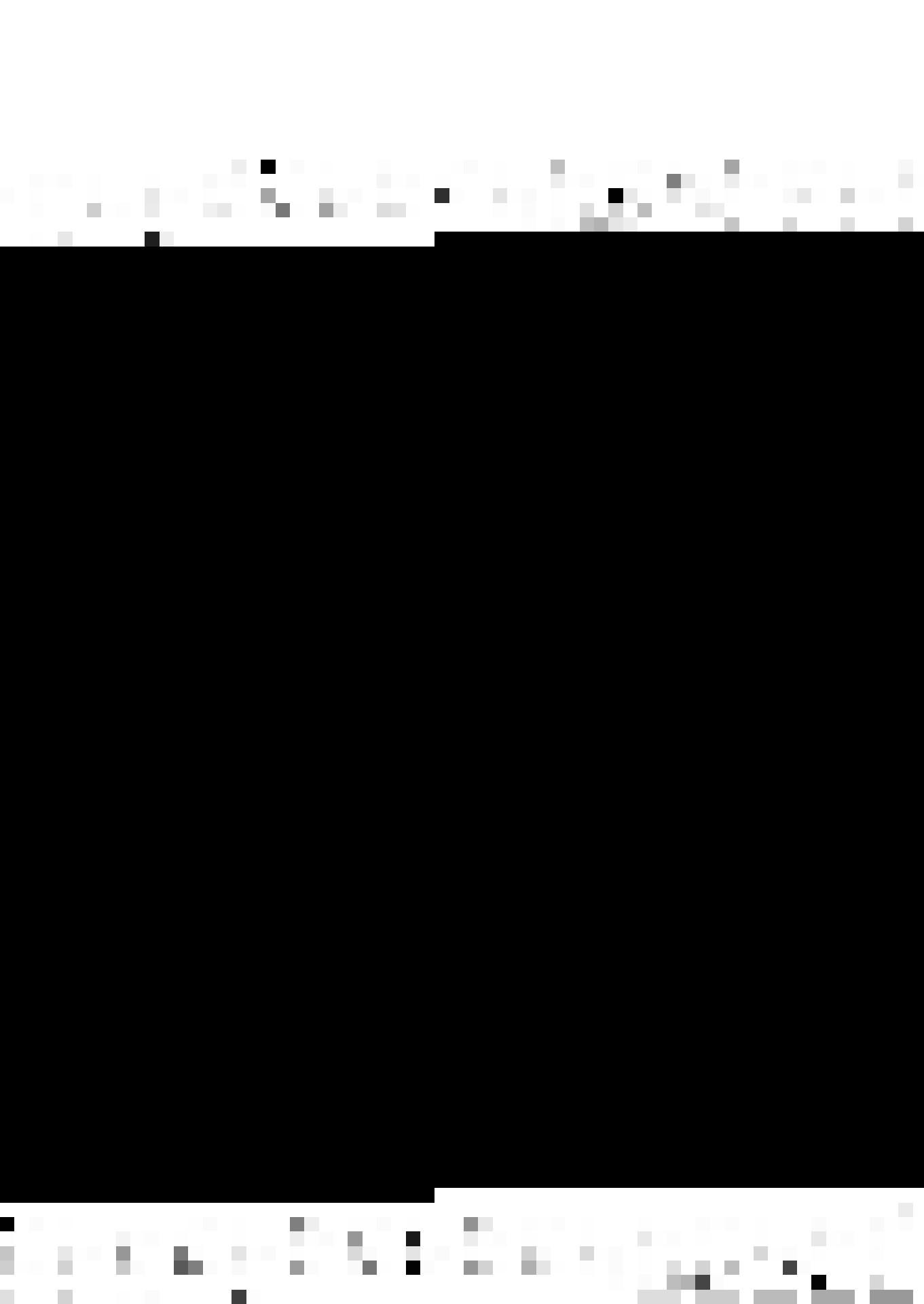
#### Lemma 3.1.1:

Let  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$ , be an nGCA. If for some  $q'$  in  $Q$ ,  $t \geq 1$  and  $z_2, \dots, z_t$  in  $\Gamma - \{z_0, z_R\}$ ,  $(q, w, \alpha z_1 \uparrow z \alpha', 0) \xrightarrow{a^*} (q', w, \alpha z_1 \dots z_t z \uparrow \alpha', 0)$ , in  $x$  moves, where  $z_1, z$  are in  $\Gamma$ , then  $x = t$ .

\*\*\*

This follows from the fact that it takes one move to cross over  $z$  and one move to cross over each symbol stored. For a proof of the lemma, see Appendix E.

- From a close look at this, we note that  $z_1$  cannot be  $z_R$ . We are not interested in the case:  $z_1 = z_R$ ; the lemma however trivially holds true for this case also in view of the fact that the stack head cannot store any stack symbol when reading  $z_R$  i.e.  $t = 1$ .



We will now introduce a few notations:

Let  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$  be an nGCA.

$(z, q, q_1, \dots, q_n)$  is an A-vector in  $M$  and

$(q, q_1, \dots, q_n, w_1, \dots, w_n)$  is a B-vector in  $M$ , where  $Z$  is in  $\Gamma$ ,  $q, q_1, \dots, q_n$  are in  $Q$  and  $w_1, \dots, w_n$  are in  $\Sigma^*$ .

Let  $(z_1, q, q_1, \dots, q_n)$  be an A-vector in  $M$  and  $(q', q'_1, \dots, q'_n, w_1, \dots, w_n)$  be a B-vector in  $M$ . We introduce a relation ' $\xrightarrow{M}$ ' between A-vectors and B-vectors in  $M$  as follows:

$$(z_1, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n)$$

iff

for some  $z_2, \dots, z_t$ ,  $t \geq 1$ ,

$$(a): (q, w, \alpha z_1 \uparrow z \alpha', 0)$$

$$\vdash^{a*} (q', w, \alpha z_1 \dots z_t z \uparrow \alpha', 0)$$

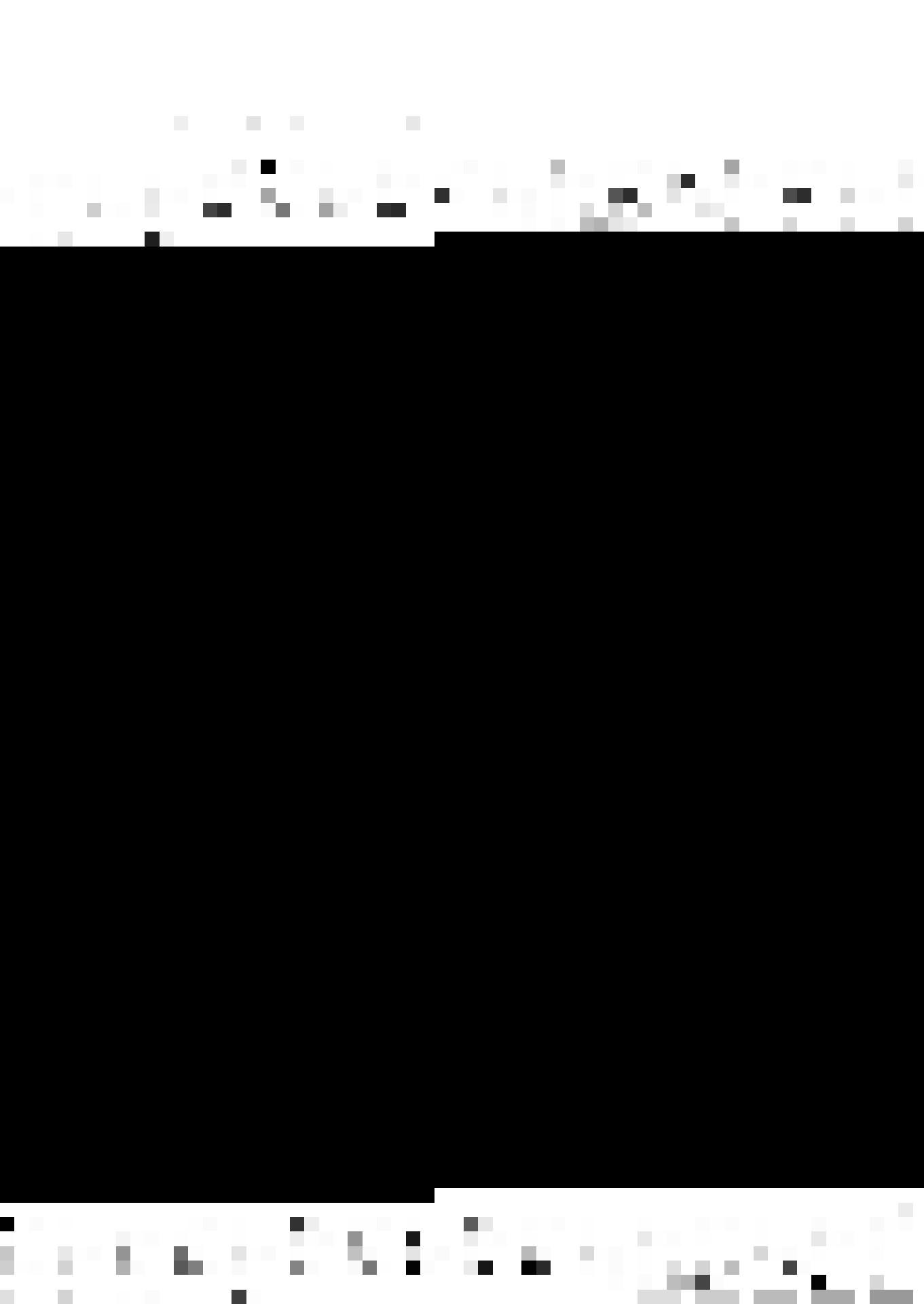
and

$$(b_j): (q_j, w_j, \alpha z_1 \uparrow z_2 \dots z_t z \alpha', j)$$

$$\vdash^{c*} (q'_j, \epsilon, \alpha z_1 \dots z_t z \uparrow \alpha', j),$$

where  $z_1, z$  are in  $\Gamma$ ,  $1 \leq j \leq n$ .

Note that the contents of the stack in the configuration shown on the right hand side of  $\vdash^{a*}$  are the same as the contents of the stack in the configuration shown on the left hand side of  $\vdash^{c*}$ , for each  $(b_j)$ .



In view of Lemma 3.1.1, the number of moves in which (a) results is the same as the number of moves in which each  $(b_j)$ ,  $1 \leq j \leq n$ , results, viz. t.

We will view the i-th move in which (a) results and the i-th move in which  $(b_j)$  results,  $1 \leq j \leq n$ , as constituting the i-th step,  $1 \leq i \leq t$ , in which,

$$(z_1, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n).$$

We will now define a normal form for nGCA's.

An nGCA  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_o, z_R, F)$  is said to be in normal form if it satisfies the following conditions:

(1) The counter is incremented in adding as well as in checking moves only when the stack head is reading  $z_R$ .

(2) A string is accepted only when the stack head is reading  $z_R$  and the counter is containing n.

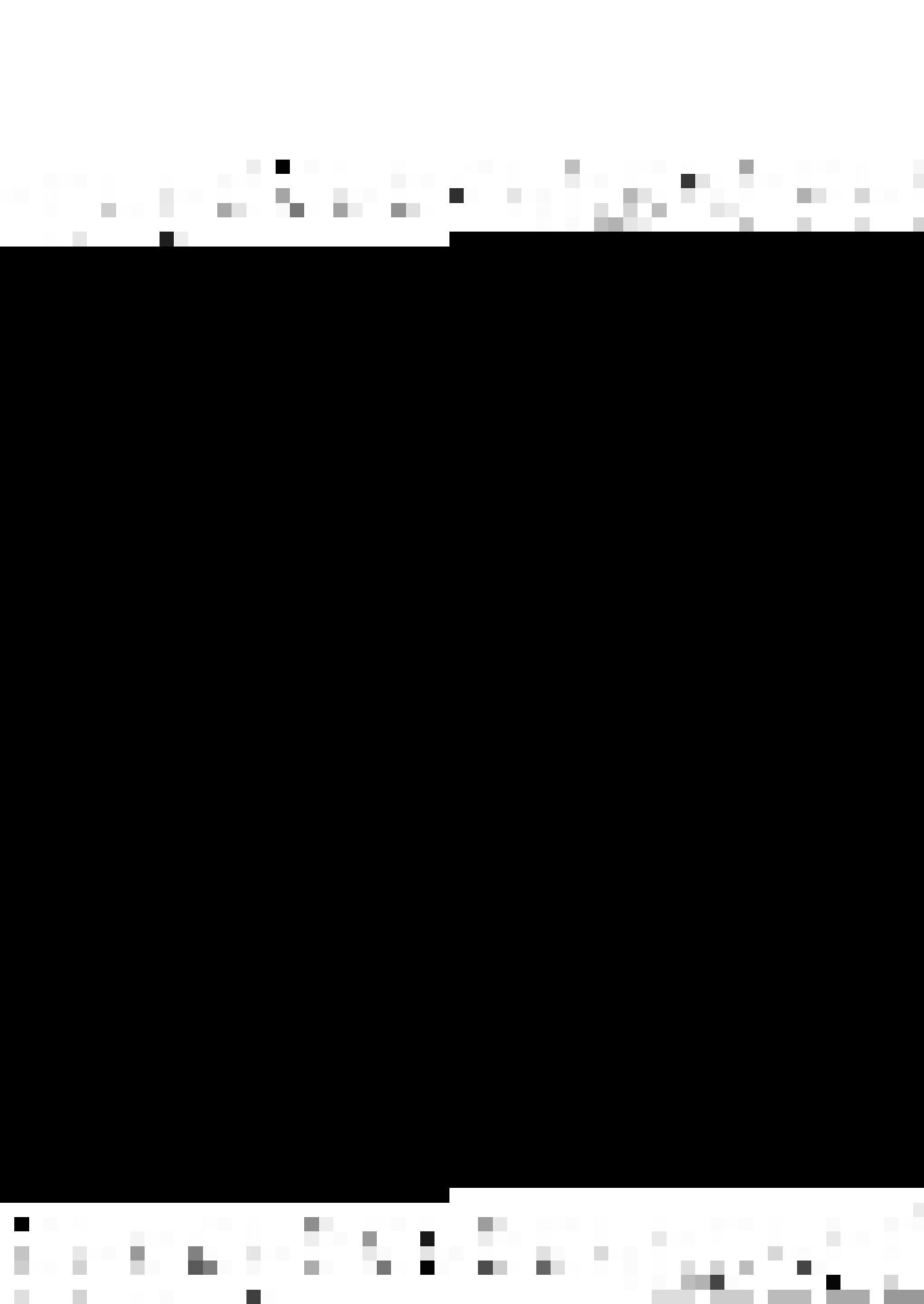
Let us now prove the normal form theorem.

Theorem 3.1.1 (Normal Form Theorem):

Given an nGCA  $M'$ , we can construct an nGCA  $M$  in normal form such that  $T(M') = T(M)$ .

Proof:

We will give the proof in two parts. For each of the two parts, we will assume an nGCA  $M' = (Q', \Sigma, \Gamma, \delta'_a, \delta'_c, C, q_0, z_o, z_R, F)$ . For part (b), we will assume that  $M'$  satisfies condition (1) of the normal form definition.



(a) Let us construct an nGCA  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_R, F)$  satisfying condition (1) of the normal form definition:

$Q = Q' \cup Q''$ . The members of  $Q''$  are those introduced below.  $Q'' = \emptyset$ , to start with.  $\delta_a$  and  $\delta_c$  comprise of those rules of  $\delta'_a$  and  $\delta'_c$ , respectively, which do not increment the counter plus those introduced below:

(i) For each  $Z$  in  $\Gamma - \{z_R\}$ ,  $q, q'$  in  $Q'$ , and  $\alpha$  in  $(\Gamma - \{z_0, z_R\})^*$ , such that  $\delta_a(q, Z) = \{(q', \alpha, 1)\}$ , introduce a new state  $q_t$  in  $Q''$  (not already in  $Q''$ ) and have:

$$\delta_a(q, Z) = \{(q_t, \alpha, 0)\},$$

$$\delta_a(q_t, z_i) = \{(q_t, \epsilon, 0)\}, \text{ for all } z_i \text{ in } \Gamma - \{z_R\}, \text{ and}$$

$$\delta_a(q_t, z_R) = \{(q', \epsilon, 1)\}.$$

(ii) For each  $Z$  in  $\Gamma - \{z_R\}$ ,  $j \geq 1$ ,  $q_j, q'_j$  in  $Q'$ ,  $a_j$  in  $\Sigma \cup \{\epsilon\}$ , such that

$\delta_c(q_j, a_j, Z, j) = \{(q'_j, 1)\}$ , introduce a new state  $q_{jt}$  in  $Q''$  (not already in  $Q''$ ) and have:

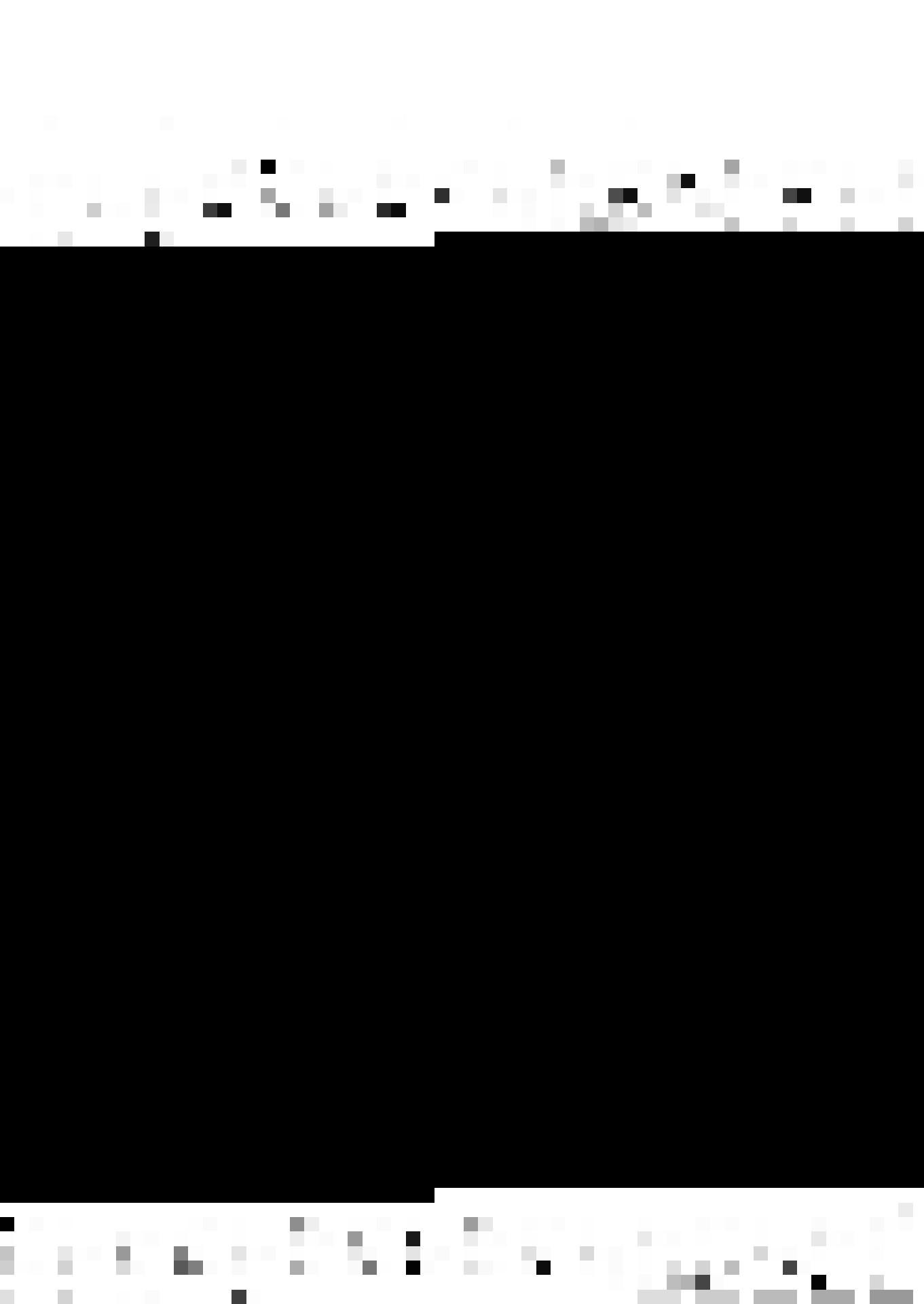
$$\delta_c(q_j, a_j, Z, j) = \{(q_{jt}, 0)\},$$

$$\delta_c(q_{jt}, \epsilon, z_i, j) = \{(q_{jt}, 0)\},$$

for all  $z_i$  in  $\Gamma - \{z_R\}$ , and

$$\delta_c(q_{jt}, \epsilon, z_R, j) = \{(q'_j, 1)\}.$$

It is easy to see that  $T(M) = T(M')$ . We omit the proof for showing this.



(b) Now, we construct an nGCA  $M = (Q, \Sigma, F, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$  in normal form, as follows:

$\delta'_a = \delta_a$ .  $Q = Q' \cup Q''$ . The states in  $Q''$  will be introduced below.  $Q'' = \emptyset$ , to start with.  $\delta_c$  is formed from  $\delta'_c$  by deleting certain rules in  $\delta'_c$  and introducing in their place some new rules, as follows:

(i) If there is a rule:

$\delta'_c(q, a, z, n) = \{(q_f, 0)\}$ , in  $\delta'_c$ , where  $z$  is in  $F - \{z_R\}$ , and  $q_f$  is in  $F$ , then delete this rule, introduce a new state  $q_{ft}$  in  $Q''$  (not already in  $Q'$ ), and define:

$$\delta_c(q, a, z, n) = \{(q_{ft}, 0)\},$$

$$\delta_c(q_{ft}, \epsilon, z_i, n) = \{(q_{ft}, 0)\},$$

for all  $z_i$  in  $F - \{z_R\}$ ,

and

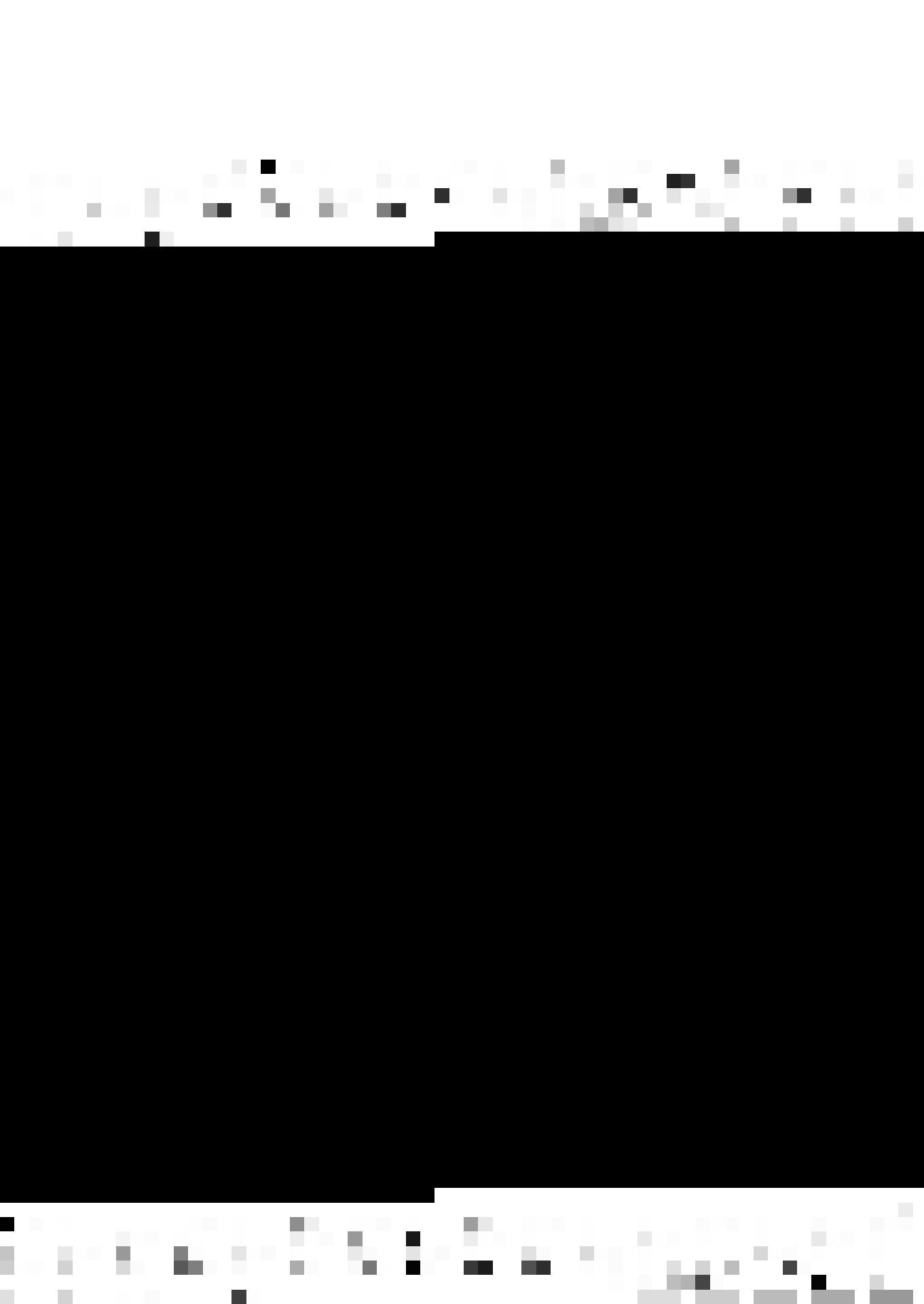
$$\delta_c(q_{ft}, \epsilon, z_R, n) = \{(q_f, 0)\}.$$

(ii) If there is a rule:

$\delta'_c(q, \epsilon, z_R, n-1) = \{(q_f, 1)\}$ , in  $\delta'_c$ , where  $q_f$  is in  $F$ , then delete this rule, introduce a new state  $q_{ft}$  in  $Q''$  (not already in  $Q'$ ), and define:

$$\delta_c(q, \epsilon, z_R, n-1) = \{(q_{ft}, 1)\},$$

$$\delta_c(q_{ft}, \epsilon, z_i, n) = \{(q_{ft}, 0)\},$$



for each  $z_i$  in  $\Gamma - \{z_R\}$ ,

and

$$\delta_C(q_{ft}, \epsilon, z_R, n) = \{(q_f, 0)\}.$$

The proof for showing that  $T(M) = T(M')$  is easy and is omitted.

Q.E.D.

An nGCA in which no adding move stores more than two stack symbols, is no less powerful than an nGCA in which we don't impose this restriction. This we shall prove in an indirect manner in section 3.2.

Let us introduce the following concept:

An nGCA is said to be of order k if there is atleast one adding move of the machine which stores k stack symbols in the stack and there is not any adding move of the machine which stores more than k stack symbols.

### 3.2 n-GENERALIZED CHECKING AUTOMATA AND n-SIMPLE COUPLED LANGUAGES :

We shall now prove the fundamental result that the class of languages accepted by nGCA is precisely the class of nSCL's. This we show by first proving that every nSCL is accepted by some nGCA and then proving that the language accepted by any nGCA is an nSCL.



Theorem 3.2.1:

Every  $(n, k)$  SCL is accepted by some nGCA  $M$  of order 2, such that  $M$  is in normal form.

Proof:

Let  $G = (V_N, V_T, P, (S)^n)$  be an  $(n, 2)$  SCG in simple normal form such that  $L(G)$  is the given  $(n, k)$  SCL. (See Cor. 2.2.14)

First, we construct an equivalent nSCG  $G'$  (equivalent in terms of the languages generated) from  $G$ , which has got just one production having the start symbol as its left hand side.

$G' = (V'_N, V_T, P', (S_1)^n)$ , where  $S_1$  is a symbol not in  $V_N \cup V_T$ ,  $V'_N = V_N \cup \{S_1\}$ , and  $P' = P \cup \{S_1 \rightarrow (S)^n\}$ . Clearly,  $L(G) = L(G')$ .

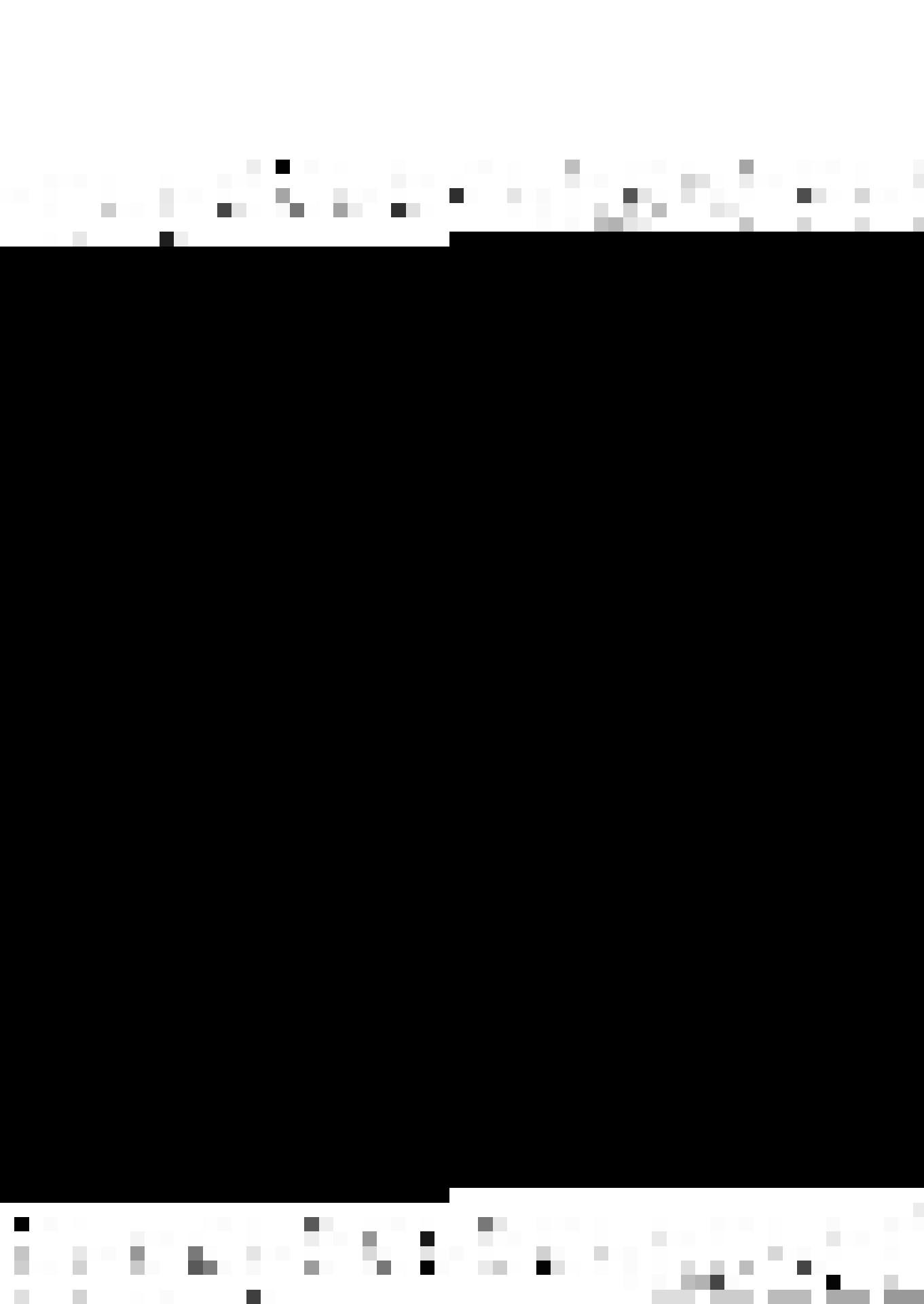
Let the productions in  $P'$  be numbered  $0, 1, \dots, m$  ( $0$  corresponding to the production with  $S_1$  as its left hand side).

Now, the required nGCA is given by:

$$M = (\{q_0, q_f\}, V_T, \{z_i \mid i = 1, \dots, m\} \cup \{z_0, z_R\}, \delta_a, \delta_c, C, q_0, z_0, z_R, \{q_f\}),$$

where  $\delta_a$  and  $\delta_c$  are specified below.

The idea behind the construction of  $M$  is the following: Let the stack symbols other than  $z_R$  represent productions in  $G'$ ,  $z_0$  representing the production number  $0$ . Let the machine have the ability to nondeterministically guess the sequence of



productions used in the leftmost derivation of a sentence in  $L(G')$ . Let this information be stored in the stack by storing the corresponding stack symbols, in order, on the stack, in the adding mode. When reading the stack symbol  $Z_i$  in the checking mode with the counter reading  $j$ , let the machine recognize the same terminal symbol (on its input tape) which  $G'$  would generate in the  $j$ -th component<sup>1</sup> of the  $i$ -th production (which may possibly be  $\epsilon$ ).

$\delta_a$  and  $\delta_c$  are specified as follows:

(a) For every pair  $(0, i)$ ,  $1 \leq i \leq m$ , of productions

$$0: S_1 \rightarrow (S)^n, \text{ and}$$

$$i: S \rightarrow (\alpha_1, \dots, \alpha_n),$$

we have,

$$\delta_a(q_0, z_0) = \{(q_0, z_i, 0)\}.$$

At the start when the stack head is reading the symbol  $z_0$ , M guesses the production used to expand  $S$  and prints the corresponding stack symbol on the stack.

(b) For every triple  $(i, j, k)$ ,  $1 \leq i, j, k \leq m$ , of productions:

$$i: A \rightarrow (A_1 A_2)^n,$$

$$j: A_1 \rightarrow (\alpha_1, \dots, \alpha_n), \text{ and}$$

$$k: A_2 \rightarrow (\beta_1, \dots, \beta_n),$$

1. Notice that every production in an nGCA has got  $n$  components in its right hand side. Here we are referring to these components.



we have,

$$\delta_a(q_0, z_i) = \{(q_0, z_j z_k, 0)\}.$$

With the stack head reading the symbol  $z_i$ , the machine guesses the productions used to expand the nonterminals  $A_1$  and  $A_2$ , say productions  $j$  and  $k$ , respectively, prints  $z_j z_k$  on the stack immediately to the right of  $z_i$  after shifting all nonblank symbols to the right of  $z_i$  by two cells and lets the stack head scan  $z_j$ .

(c) For every nonterminating production  $i$ ,  $0 \leq i \leq m$ , we have,

$$\delta_c(q_0, \epsilon, z_i, j) = \{(q_0, 0)\}, j = 1, \dots, n.$$

Since the grammar  $G'$  is in simple normal form, there is no terminal symbol on the right hand side of a nonterminating production. In the checking mode with the counter containing  $j$ ,  $1 \leq j \leq n$ , the machine likewise does not move its input head when the stack head is reading a symbol corresponding to a nonterminating production.

(d) For every terminating production  $i$ ,  $0 \leq i \leq m$ ,

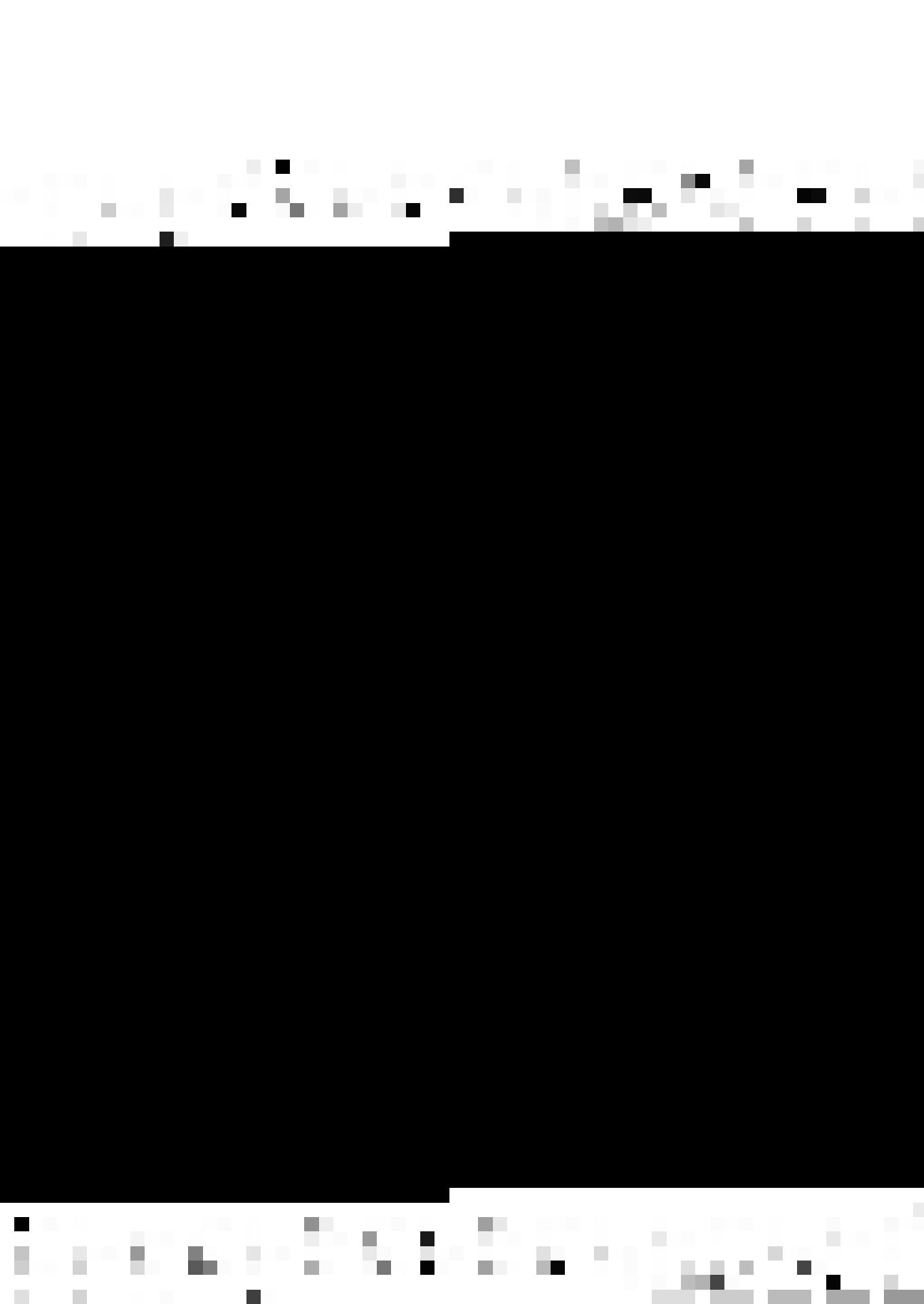
$$i: A \rightarrow (a_1, \dots, a_n),$$

we have:

$$\delta_a(q_0, z_i) = \{(q_0, \epsilon, 0)\},$$

and

$$\delta_c(q_0, a_j, z_i, j) = \{(q_0, 0)\}, j = 1, \dots, n.$$



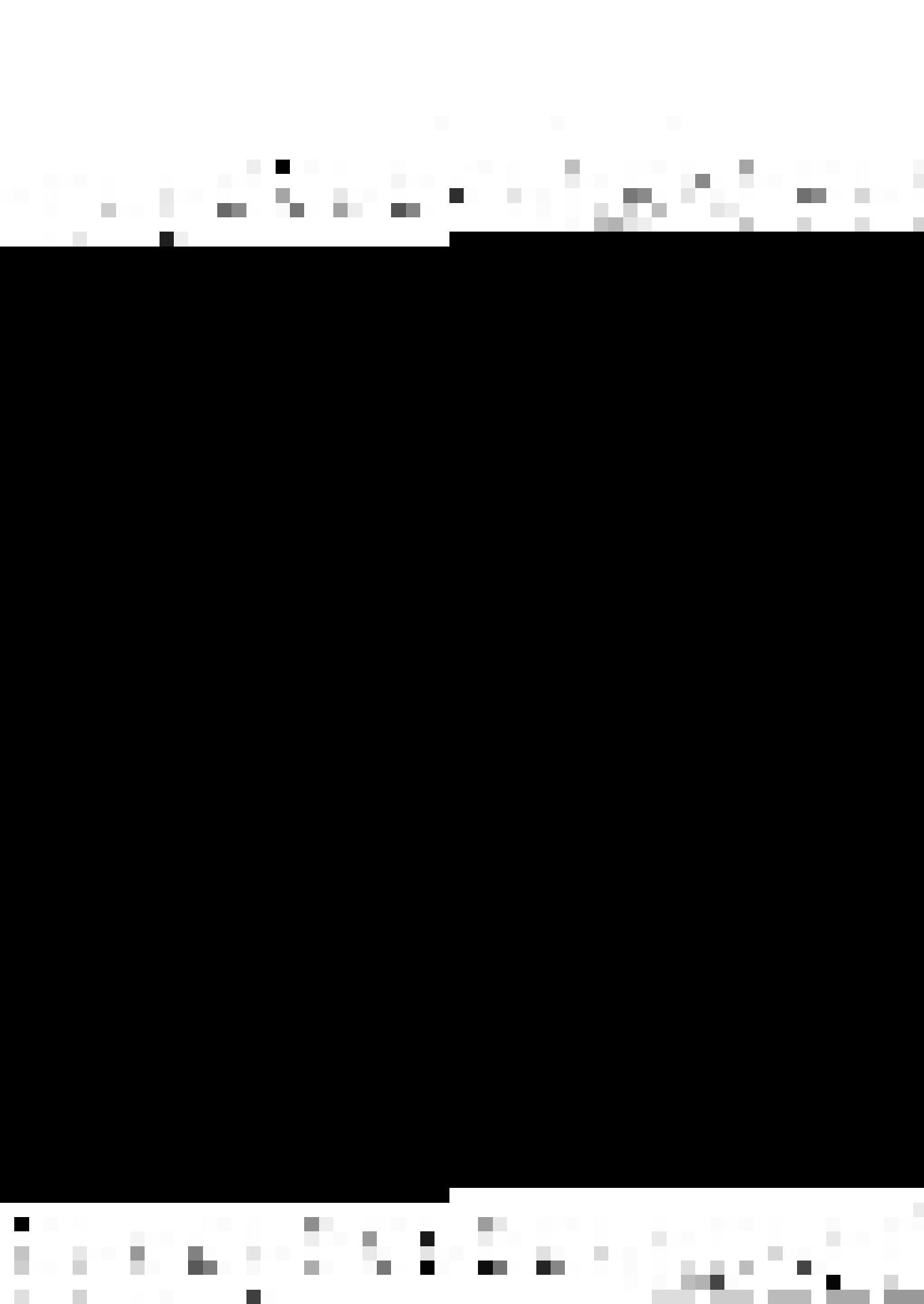
In the adding mode, when the stack head is reading a symbol  $Z_i$  which corresponds to the terminating production  $i$ , it simply moves right by one cell without storing anything on the stack, which is in keeping with the idea that in the adding mode  $M$  must store symbols on the stack which indicate the productions used in a leftmost derivation of a sentence in  $L(G')$ . Again, in the checking mode, while reading  $Z_i$ , with the counter containing  $j$ ,  $1 \leq j \leq n$ , the machine recognizes that symbol (possibly  $\epsilon$ ) on the input tape which comprises the  $j$ -th component of the right hand side of the production  $i$ .

$$(e) \quad \delta_a(q_0, Z_R) = \{(q_0, \epsilon, 1)\},$$

$$\delta_c(q_0, \epsilon, Z_R, j) = \{(q_0, 1)\}, \quad j = 1, \dots, n-1, \text{ and}$$

$$\delta_c(q_0, \epsilon, Z_R, n) = \{(q_f, 0)\}.$$

When the production sequence used for some leftmost derivation has been guessed by the machine and the corresponding stack symbols stored on the stack in correct order, in the adding mode, it must switch over to the checking mode. This it does by reading  $Z_R$ . Similarly, in the checking mode with the counter containing  $j$ ,  $1 \leq j \leq n-1$ , after all the terminal symbols have been recognized in the correct order, which are generated in the  $j$ -th component of the productions used in the leftmost derivation, the machine must increment its counter, which it does while reading  $Z_R$ . Again after the stack head has checked terminal symbols against symbols in the stack in the  $n$ -th such scan, it must go into the final state.



Observe that  $M$  is of order 2 and is in normal form.

Now, we proceed to prove that  $T(M) = L(G')$ .

We will first make some statements, which follow from the above construction.

- (1) Using the production  $p$ ,  $a_j$  will be generated by  $G'$  in the  $j$ -th component iff

using the rule:  $\delta_c(q_0, a_j, z_p, j) = \{(q_0, 0)\}$ ,  
 $M$  recognizes  $a_j$ , when the state of the machine is  $q_0$ , the stack head is reading  $z_p$ , and the counter contains  $j$ , where  $1 \leq j \leq n$ ,  $a_j$  is in  $V_T \cup \{\epsilon\}$ , and  $0 \leq p \leq m$ .

This follows from (d).

Let ' $i_1, \dots, i_j$ ' is a terminating sequence of productions in  $G'$ , denote the fact that for some  $A$  in  $V_N$ , and  $x_i$ ,  $1 \leq i \leq n$ , in  $V_T^*$ ,

$$(A)^n \xrightarrow[G']{*} (x_1, \dots, x_n),$$

by a leftmost derivation, using the production sequence  $i_1, \dots, i_j$

Note that  $(x_1, \dots, x_n)$  as well as the nonterminal expanded at each step of the leftmost derivation get uniquely defined once the terminating sequence of productions is specified.



(2)  $i_1, \dots, i_j$  is a terminating sequence of productions in  $G'$

iff

$$(q_0, w, y' z_{i_1} \uparrow z_y, 0)$$

$$\vdash_M^{a^*} (q_0, w, y' z_{i_1} \dots z_{i_j} z \uparrow y, 0),$$

where  $j \geq 1$ , and  $z_{i_1}, z$  are in  $\Gamma$ .

The above statement may be elaborated as: Let the stack head be reading the symbol  $z_{i_1}$ . Let a stack symbol  $z$  be immediately to the right of  $z_{i_1}$  in the stack. Let the machine be in state  $q_0$  and the counter contain zero. If  $i_1, \dots, i_j$  is a terminating sequence of productions in  $G'$  then through a sequence of adding moves the string  $z_{i_2} \dots z_{i_j}$  gets printed between  $z_{i_1}$  and  $z$ , and the stack head moves right by  $j$  cells and starts reading the symbol  $z$ , the machine state as well as the content of the counter remaining the same, and vice versa.

For a proof of the above statement, see Appendix F.

The following two statements follow directly from the construction of  $M$ :

(3) The counter is incremented in the adding as well as the checking moves when and only when the stack head is reading  $z_R$  and the machine is in state  $q_0$ .



(4) The machine enters the final state when and only when the stack head is reading,  $Z_R$ , the counter is containing  $n$  and the machine is in state  $q_0$ .

Now, in order to prove that  $T(M) = L(G')$ , it suffices (in view of the manner  $M$  is constructed, particularly noting (3) and (4)) to prove that:

$(S_1)^n \xrightarrow[G']{*} (x_1, \dots, x_n)$ , where  $x_i$ ,  $1 \leq i \leq n$ , is in  $V_T^*$ ,

by a leftmost derivation using the production sequence

$0, i_1, \dots, i_j$

iff

$(q_0, x_1 \dots x_n, z_0 \uparrow z_R, 0)$

$\vdash_M^{a*} (q_0, x_1 \dots x_n, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, 0)$

$\vdash_M^a (q_0, x_1 \dots x_n, z_0 \uparrow z_{i_1} \dots z_{i_j} z_R, 1)$

$\vdash_M^{c*} (q_0, x_2 \dots x_n, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, 1)$

$\vdash_M^c (q_0, x_2 \dots x_n, z_0 \uparrow z_{i_1} \dots z_{i_j} z_R, 2)$

.

$\vdash_M^{c*} (q_0, x_n, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, n-1)$

$\vdash_M^c (q_0, x_n, z_0 \uparrow z_{i_1} \dots z_{i_j} z_R, n)$

$\vdash_M^{c*} (q_0, \epsilon, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, n)$

$\vdash_M^c (q_f, \epsilon, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, n)$



only if

It follows from (2) that  $(q_0, x_1 \dots x_n, z_0 \uparrow z_R, 0)$   $\xrightarrow[M]{a^*} (q_0, x_1 \dots x_n, z_0 z_{i_1} \dots z_{i_j} z_R \uparrow, 0)$ . The required result follows now in view of (3), (4) and the 'only if' part of (1).

if

It follows from (2) that  $0, i_1, \dots, i_j$  is a terminating sequence of productions which means that  $(S_1)^n \xrightarrow[G]{*} (z_1, \dots, z_n)$ , where  $z_i, 1 \leq i \leq n$ , is in  $V_T^*$ , by a leftmost derivation using the production sequence  $0, i_1, \dots, i_j$ . But in view of the 'if' part of (1),  $z_i = x_i, 1 \leq i \leq n$ .

Q.E.D.

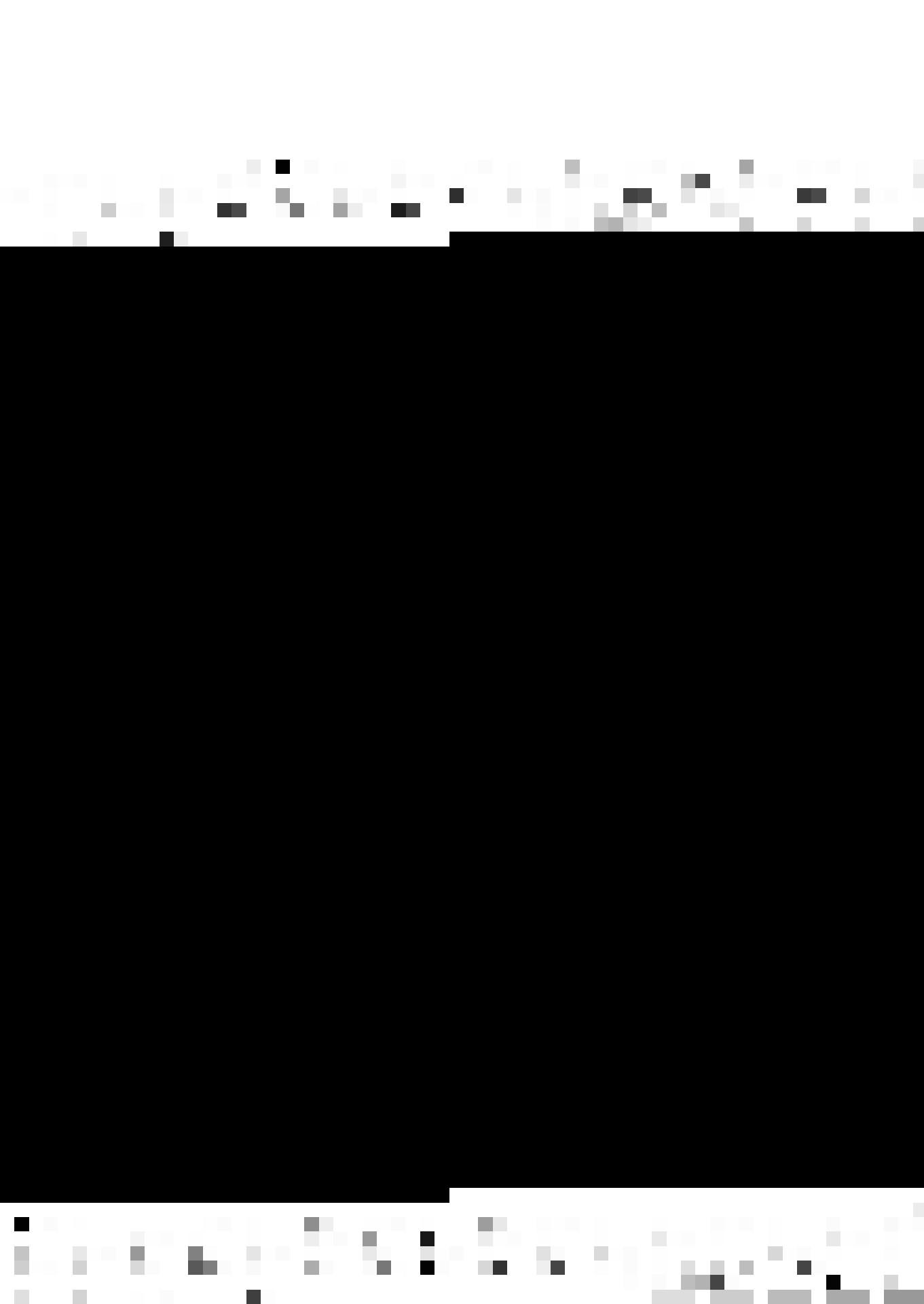
To illustrate the proof of the above theorem, let us consider an example:

Example 3.2.1:

$$G = (\{S, S', A, B, C\}, \{a, b, c\}, P, (S)^2),$$

where P comprises of the productions:

1.  $S \rightarrow (AB)^2,$
2.  $S \rightarrow (CB)^2,$
3.  $S' \rightarrow (AB)^2,$
4.  $S' \rightarrow (CB)^2,$
5.  $A \rightarrow (CS')^2,$
6.  $C \rightarrow (a, c)$ , and
7.  $B \rightarrow (b, \epsilon),$



is a 2SCG in simple normal form generating the language  
 $\{a^i b^i c^i \mid i \geq 1\}.$

We shall use the construction given in the proof of Theorem 3.2.1 to construct a 2GCA  $M$  such that  $T(M) = L(G)$ . (We shall be following the steps of the construction for constructing  $M$ ).

$$G' = (\{S_1, S, S', A, B, C\}, \{a, b, c\}, P', (S_1)^2),$$

where

$P'$  comprises of the following productions:

0.  $S_1 \rightarrow (S)^2,$
1.  $S \rightarrow (AB)^2,$
2.  $S \rightarrow (CB)^2,$
3.  $S' \rightarrow (AB)^2,$
4.  $S' \rightarrow (CB)^2,$
5.  $A \rightarrow (CS')^2,$
6.  $C \rightarrow (a, c),$  and
7.  $B \rightarrow (b, \epsilon).$

The required 2GCA is given by:

$$M = (\{q_0, q_f\}, \{a, b, c\}, \{z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_R\}, \delta_a, \delta_c, C, q_0, z_0, z_R, \{q_f\}),$$

where  $\delta_a$  and  $\delta_c$  are specified as follows:



- (a) (i)  $\delta_a(q_0, z_0) = \{(q_0, z_1, 0)\}$ ; and  
(ii)  $\delta_a(q_0, z_0) = \{(q_0, z_2, 0)\}.$
- (b) (i)  $\delta_a(q_0, z_1) = \{(q_0, z_5 z_7, 0)\},$   
(ii)  $\delta_a(q_0, z_2) = \{(q_0, z_6 z_7, 0)\},$   
(iii)  $\delta_a(q_0, z_3) = \{(q_0, z_5 z_7, 0)\},$   
(iv)  $\delta_a(q_0, z_4) = \{(q_0, z_6 z_7, 0)\},$   
(v)  $\delta_a(q_0, z_5) = \{(q_0, z_6 z_3, 0)\},$  and  
(vi)  $\delta_a(q_0, z_5) = \{(q_0, z_6 z_4, 0)\}.$
- (c) (i)  $\delta_c(q_0, \epsilon, z_0, j) = \{(q_0, 0)\},$   
(ii)  $\delta_c(q_0, \epsilon, z_1, j) = \{(q_0, 0)\},$   
(iii)  $\delta_c(q_0, \epsilon, z_2, j) = \{(q_0, 0)\},$   
(iv)  $\delta_c(q_0, \epsilon, z_3, j) = \{(q_0, 0)\},$   
(v)  $\delta_c(q_0, \epsilon, z_4, j) = \{(q_0, 0)\},$   
(vi)  $\delta_c(q_0, \epsilon, z_5, j) = \{(q_0, 0)\},$   
 $j = 1, 2.$
- (d) (i)  $\delta_a(q_0, z_6) = \{(q_0, \epsilon, 0)\},$   
(ii)  $\delta_a(q_0, z_7) = \{(q_0, \epsilon, 0)\},$   
(iii)  $\delta_c(q_0, a, z_6, 1) = \{(q_0, 0)\},$   
(iv)  $\delta_c(q_0, b, z_7, 1) = \{(q_0, 0)\},$



$$(v) \delta_c(q_0, c, z_6, 2) = \{(q_0, 0)\}, \text{ and}$$

$$(vi) \delta_c(q_0, \epsilon, z_7, 2) = \{(q_0, 0)\},$$

$$(e) (i) \delta_a(q_0, z_R) = \{(q_0, \epsilon, 1)\},$$

$$(ii) \delta_c(q_0, \epsilon, z_R, 1) = \{(q_0, 1)\}, \text{ and}$$

$$(iii) \delta_c(q_0, \epsilon, z_R, 2) = \{(q_f, 0)\}.$$

It is easy to see that  $T(M) = \{a^i b^i c^i \mid i \geq 1\}.$

\*\*\*

Now, let us prove that the language accepted by any nGCA is an nSCL.

### Theorem 3.2.2:

If  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$  is an nGCA, then there is an nSCG  $G$  such that  $T(M) = L(G).$

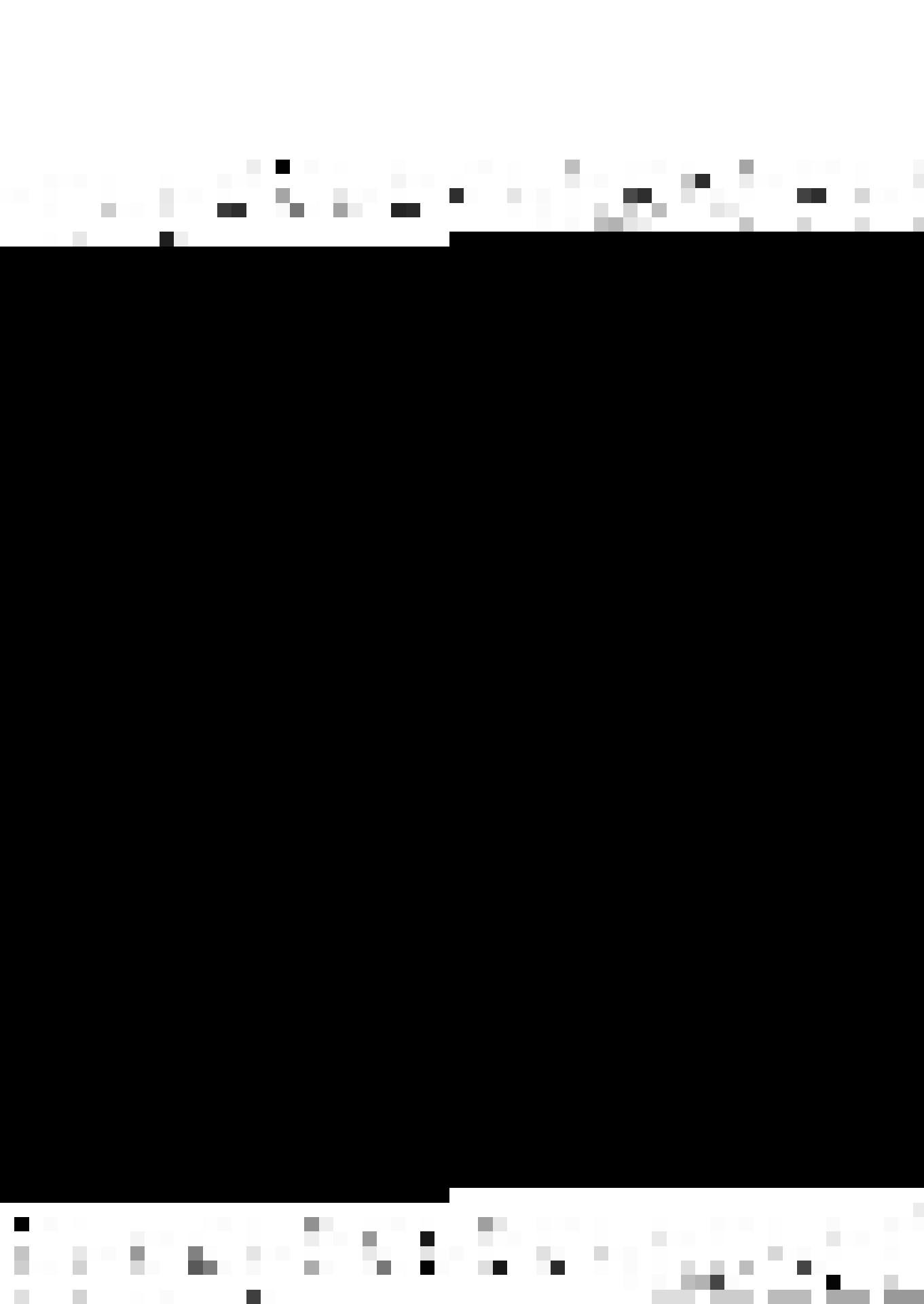
### Proof:

Let  $M$  be in normal form. We shall construct an nSCG  $G = (V_N, \Sigma, P, (S)^n)$ , such that  $T(M) = L(G)$ , as follows:

$V_N$  is the set of all objects of the form

$[q, q_1, \dots, q_n, z, q'_1, \dots, q'_n]$ , where  $q, q', q_i, q'_i$ ,  $1 \leq i \leq n$ , are in  $Q$  and  $z$  is in  $\Gamma - \{z_R\}$ , plus a new symbol  $S$ .

The idea is to let  $[q, q_1, \dots, q_n, z, q'_1, q'_2, \dots, q'_n]$  contain the information that for some  $w_1, \dots, w_n$  in  $\Sigma^*$ ,



$(z, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n)$ .

Note that if

$(z_0, q_0, p_1, \dots, p_n) \xrightarrow{M} (p', p'_1, \dots, p'_n, x_1, \dots, x_n)$ ,

for some  $p_1, \dots, p_n, p', p'_1, \dots, p'_n$  in Q,  $x_1, \dots, x_n$  in  $\Sigma^*$ ,

then  $x_1 \dots x_n$  is in  $T(M)$  only if there are proper end rules i.e. if

$$\delta_a(p', z_R) = \{(p_1, \epsilon, 1)\},$$

$$\delta_c(p'_i, \epsilon, z_R, i) = \{(p_{i+1}, 1)\}, 1 \leq i \leq n-1, \text{ and}$$

$$\delta_c(p'_n, \epsilon, z_R, n) = \{(q_f, 0)\}, \text{ where } q_f \text{ is in F.}$$

This aspect is taken care of by providing the nonterminal S.

The idea behind the construction of all the productions except those which take care of the 'end' rules is to view the acceptance of a certain string as partly comprising of steps in which for some  $q, q_1, \dots, q_n, q', q'_1, \dots, q'_n$  in Q

$(z_0, q_0, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n)$

such that  $w = w_1 \dots w_n$ .

Formally, P is constructed as follows:

1.  $S \rightarrow ([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n])^n$  is in P

if

$$\delta_a(q', z_R) = \{(q_1, \epsilon, 1)\},$$

$$\delta_c(q'_j, \epsilon, z_R, j) = \{(q_{j+1}, 1)\}, 1 \leq j \leq n-1, \text{ and}$$

$$\delta_c(q'_n, \epsilon, z_R, n) = \{(q_f, 0)\}, \text{ where } q_f \text{ is in F.}$$



2. For every  $q'$ ,  $q'_1, \dots, q'_n$ ,  ${}^2q$ ,  ${}^2q_1, \dots, {}^2q_n$ ,  $\dots, {}^m_q$ ,  
 ${}^m_{q_1}, \dots, {}^m_{q_n}$  in  $Q$ ,  $m \geq 1$ ,

$$[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n] \rightarrow (a_1 [{}^1q, {}^1q_1, \dots, {}^1q_n, z_1, {}^2q, {}^2q_1, \dots, {}^2q_n]$$

$$[{}^2q, {}^2q_1, \dots, {}^2q_n, z_2, {}^3q, {}^3q_1, \dots, {}^3q_n] \dots$$

$$\dots [{}^m_q, {}^m_{q_1}, \dots, {}^m_{q_n}, z_m, q', q'_1, \dots, q'_n], \dots$$

$$\dots, a_n [{}^1q, {}^1q_1, \dots, {}^1q_n, z_1, {}^2q, {}^2q_1, \dots, {}^2q_n]$$

$$[{}^2q, {}^2q_1, \dots, {}^2q_n, z_2, {}^3q, {}^3q_1, \dots, {}^3q_n] \dots$$

$$\dots [{}^m_q, {}^m_{q_1}, \dots, {}^m_{q_n}, z_m, q', q'_1, \dots, q'_n]),$$

is in  $P$

if

$$\delta_a(q, z) = \left\{({}^1q, z_1 \dots z_m, 0)\right\}, \text{ and}$$

$$\delta_c(q_j, a_j, z, j) = \left\{({}^1q_j, 0)\right\}, \quad 1 \leq j \leq n.$$

3.  $[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n] \rightarrow (a_1, \dots, a_n),$

is in  $P$

if

$$\delta_a(q, z) = \left\{(q', \epsilon, 0)\right\}, \text{ and}$$

$$\delta_c(q_j, a_j, z, j) = \left\{(q'_j, 0)\right\}, \quad 1 \leq j \leq n.$$

Now, in order to prove that  $T(\mathbb{M}) = L(G)$ , we shall first prove that for any  $q, q_1, \dots, q_n, q', q'_1, \dots, q'_n$  in  $Q$ ,  $z$  in  $\Gamma - \{z_R\}$ ,  $w_1, \dots, w_n$  in  $\Sigma^*$ ,



$$A: ([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{*} (w_1, \dots, w_n)$$

iff

$$(z, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n).$$

if

Suppose that  $(z, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n)$  in  $j$  steps. We shall prove that

$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{*} (w_1, \dots, w_n),$$

by induction on  $j$ .

Suppose that  $j = 1$ . This implies that,

$$\delta_a(q, z) = \{(q', e, o)\}, \text{ and}$$

$$\delta_c(q_j, w_j, z, j) = \{(q'_j, o)\}, 1 \leq j \leq n.$$

From the construction of  $G$ , it then follows that

$$[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n]$$

$\rightarrow (w_1, \dots, w_n)$ , is in  $P$ , so that

$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{*} (w_1, \dots, w_n).$$



Suppose as an induction hypothesis that the 'if' part of A is true if  $j \leq k - 1$  for some  $k \geq 2$ .

Now, assume that  $j = k$ . Let in the first step, the rules used be:

$$\left. \begin{array}{l} \delta_a(q, z) = \left\{ \left( {}^1 q, z_1 \dots z_m, 0 \right) \right\}, \text{ and} \\ \delta_c(q_j, a_j, z, j) = \left\{ \left( {}^1 q_j, 0 \right) \right\}, 1 \leq j \leq n, \text{ for} \\ \text{some } {}^1 q, {}^1 q_j, 1 \leq j \leq n, \text{ in } Q, z_1, \dots, z_m \text{ in} \\ \Gamma - \left\{ z_0, z_R \right\}, m \geq 1 \text{ and } a_j \text{ in } \Sigma \cup \{\epsilon\}. \end{array} \right\} \quad (I)$$

Now, it must be that  $w_i$ ,  $1 \leq i \leq n$ , can be written as  $a_i x_{i1} \dots x_{im}$ , such that

$$(z_1, {}^1 q, {}^1 q_1, \dots, {}^1 q_n) \xrightarrow{M} ({}^2 q, {}^2 q_1, \dots, {}^2 q_n, x_{11}, \dots, x_{n1}).$$

$$(z_2, {}^2 q, {}^2 q_1, \dots, {}^2 q_n) \xrightarrow{M} ({}^3 q, {}^3 q_1, \dots, {}^3 q_n, x_{12}, \dots, x_{n2}),$$

•

•

$$(z_m, {}^m q, {}^m q_1, \dots, {}^m q_n) \xrightarrow{M} ({}^q, {}^q_1, \dots, {}^q_n, x_{1m}, \dots, x_{nm})$$

each in fewer than  $k$  steps, for some  ${}^2 q, {}^2 q_1, \dots, {}^2 q_n, \dots, {}^m q, {}^m q_1, \dots, {}^m q_n$ , in  $Q$  (see how the relation  $\xrightarrow{M}$  is defined between A-vectors and B-vectors in  $M$ ).

Using the induction hypothesis, we have each of the following:

$$({}^1 q, {}^1 q_1, \dots, {}^1 q_n, z_1, {}^2 q, {}^2 q_1, \dots, {}^2 q_n)^n$$

$$\xrightarrow[G]{*} (x_{11}, \dots, x_{n1}),$$



$$([^2q, ^2q_1, \dots, ^2q_n, z_2, ^3q, ^3q_1, \dots, ^3q_n])^n$$

$$\xrightarrow[G]{*} (x_{12}, \dots, x_{n2}),$$

•

•

$$([^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{*} (x_{1m}, \dots, x_{nm}).$$

Also, from (I) and the construction of  $G$ , it follows that

$$[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n]$$

$$\rightarrow (a_1 [^1q, ^1q_1, \dots, ^1q_n, z_1, ^2q, ^2q_1, \dots, ^2q_n]$$

$$[^2q, ^2q_1, \dots, ^2q_n, z_2, ^3q, ^3q_1, \dots, ^3q_n] \dots$$

$$\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n],$$

$$\dots, a_n [^1q, ^1q_1, \dots, ^1q_n, z_1, ^2q, ^2q_1, \dots, ^2q_n]$$

$$[^2q, ^2q_1, \dots, ^2q_n, z_2, ^3q, ^3q_1, \dots, ^3q_n] \dots$$

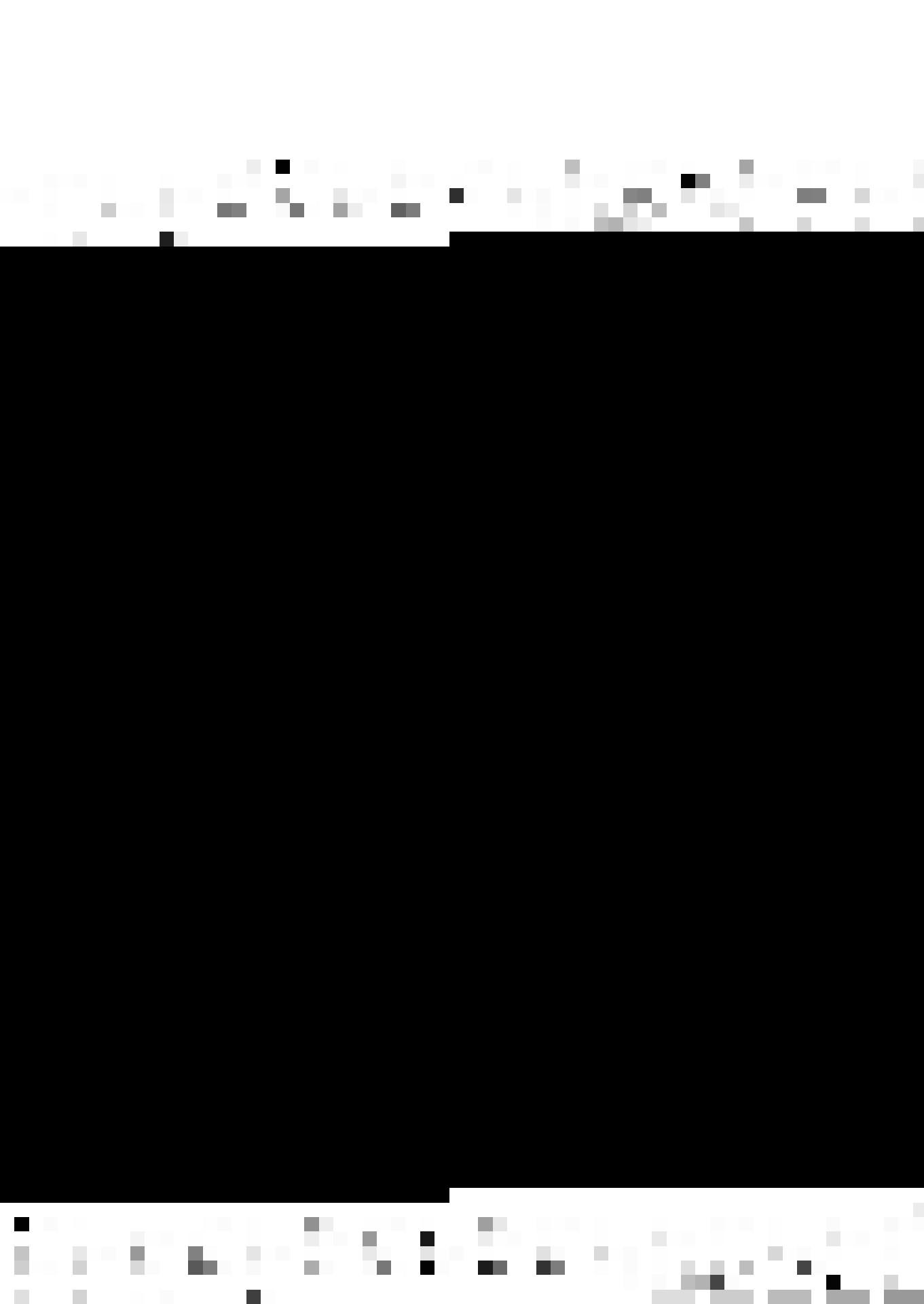
$$\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n]),$$

is in  $P$ .

Thus, it follows from above that

$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{*} (w_1, \dots, w_n)$$



$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n \xrightarrow[G]{*} (w_1, \dots, w_n)$$

by a j-step leftmost derivation. We shall prove that

$$(z, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n), \text{ by induction on } j.$$

Suppose that  $j = 1$ . This implies that the production

$[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n] \rightarrow (w_1, \dots, w_n)$ , is in P. This in turn implies (in view of the way G is constructed), the existence of the rules:

$$\delta_a(q, z) = \{(q', \epsilon, 0)\}, \text{ and}$$

$$\delta_c(q_j, w_j, z, j) = \{(q'_j, 0)\}, 1 \leq j \leq n,$$

$$\text{so that } (z, q, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n).$$

As an induction hypothesis, suppose that the 'only if' part of A is true for  $j$  (the number of steps in which  $([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n \xrightarrow[G]{*} (w_1, \dots, w_n)$  by a leftmost derivation) less than or equal to  $k-1$ ,  $k \geq 2$ .

Now, assume that  $j = k$ . Let the first step of the leftmost derivation by which

$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n \xrightarrow[G]{*} (w_1, \dots, w_n),$$

be:

$$([q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n])^n$$

$$\xrightarrow[G]{} (a_1 [^1q, ^1q_1, \dots, ^1q_n, z_1, ^2q, ^2q_1, \dots, ^2q_n] \dots$$

$$[^2q, ^2q_1, \dots, ^2q_n, z_2, ^3q, ^3q_1, \dots, ^3q_n] \dots$$



$\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n], \dots$   
 $\dots, a_n [^1 q, ^1 q_1, \dots, ^1 q_n, z_1, ^2 q, ^2 q_1, \dots, ^2 q_n]$   
 $[^2 q, ^2 q_1, \dots, ^2 q_n, z_2, ^3 q, ^3 q_1, \dots, ^3 q_n] \dots$   
 $\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n]),$

for some  $a_i$ ,  $1 \leq i \leq n$ , in  $\Sigma \cup \{\epsilon\}$ ,  $m \geq 1$ ,  $^1 q, ^1 q_1, \dots, ^1 q_n, \dots$   
 $\dots, ^m q, ^m q_1, \dots, ^m q_n$  in  $Q$ , which obviously uses the production:

$[q, q_1, \dots, q_n, z, q', q'_1, \dots, q'_n]$   
 $\rightarrow (a_1 [^1 q, ^1 q_1, \dots, ^1 q_n, z_1, ^2 q, \dots, ^2 q_n]$   
 $[^2 q, ^2 q_1, \dots, ^2 q_n, z_2, ^3 q, ^3 q_1, \dots, ^3 q_n] \dots$   
 $\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n], \dots$   
 $\dots, a_n [^1 q, ^1 q_1, \dots, ^1 q_n, z_1, ^2 q, ^2 q_1, \dots, ^2 q_n]$   
 $[^2 q, ^2 q_1, \dots, ^2 q_n, z_2, ^3 q, ^3 q_1, \dots, ^3 q_n] \dots$   
 $\dots [^m q, ^m q_1, \dots, ^m q_n, z_m, q', q'_1, \dots, q'_n]).$

(Note from the construction of  $G$  that there is no other type of a nonterminating production which could be used).

It then follows from the construction of  $G$  that:

$$\left. \begin{array}{l} \delta_a(q, z) = \left\{ (^1 q, z_1, \dots, z_m, 0) \right\}, \text{ and} \\ \delta_c(q_j, a_j, z, j) = \left\{ (^1 q_j, 0) \right\}, 1 \leq j \leq n \end{array} \right\} \text{II}$$

Now, it must be that  $w_i$  can be written as  $a_i x_{i1} \dots$

$\dots x_{im}$ ,  $1 \leq i \leq n$ , such that,



$([i_q, i_{q_1}, \dots, i_{q_n}, z_i, i+1_q, i+1_{q_1}, \dots, i+1_{q_n}]^n)$

 $\xrightarrow[G]{*} (x_{1i}, \dots, x_{ni}), 1 \leq i \leq m, \text{ by a leftmost derivation of fewer than } k \text{ steps, where } i+1_q = q', i+1_{q_1} = q'_1, \dots, i+1_{q_n} = q'_n, 1 \leq l \leq n.$ 

Using the induction hypothesis, we have:

$$(z_i, i_q, i_{q_1}, \dots, i_{q_n})$$

$$\xrightarrow[M]{} (i+1_q, i+1_{q_1}, \dots, i+1_{q_n}, x_{1i}, \dots, x_{ni}),$$

$1 \leq i \leq m-1$ , and

$$(z_m, i_q, i_{q_1}, \dots, i_{q_n}) \xrightarrow[M]{} (q', q'_1, \dots, q'_n, x_{1m}, \dots$$

$\dots, x_{nm}).$

Combining this with (II), we have;

$$(z, q, q_1, \dots, q_n) \xrightarrow[M]{} (q', q'_1, \dots, q'_n, w_1, \dots, w_n).$$

This completes the proof of A.

Now, let us come to the actual problem of proving that  $T(M) = L(G)$ .

Suppose that

$$(S)^n \xrightarrow[G]{} ([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n]^n)$$

$$\xrightarrow[G]{*} (w_1, \dots, w_n), \text{ for some } q_1, \dots, q_n, q', q'_1, \dots$$

$\dots, q'_n$  in  $Q$ , by a leftmost derivation, where  $w_i, 1 \leq i \leq n$ , is in  $\Sigma^*$ . This implies that the production used in the first step of the derivation is:

$$S \rightarrow ([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n])^n$$

(Observe from the construction of G that only this type of production can be used in the first step of the derivation).

From the construction of G, it follows then that

$$\left. \begin{aligned} \delta_a(q', z_R) &= \{(q_1, \epsilon, 1)\}, \\ \delta_c(q'_j, \epsilon, z_R, j) &= \{(q_{j+1}, 1)\}, \quad 1 \leq j \leq n-1, \text{ and} \\ \delta_c(q'_n, \epsilon, z_R, n) &= \{(q_f, 0)\}, \end{aligned} \right\} \text{ III}$$

where  $q_f$  is in F.

Again, it follows from above that

$$\begin{aligned} ([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n])^n \\ \xrightarrow[G]{*} (w_1, \dots, w_n). \end{aligned}$$

Using A, this implies that

$$(z_0, q_0, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, w_1, \dots, w_n).$$

This along with (III), leads to the conclusion that  $w_1, \dots, w_n$  is in  $T(M)$ .

Again, let w be in  $T(M)$ . This (noting that M is in normal form) implies that for some  $q_1, \dots, q_n, q', q'_1, \dots, q'_n$  in Q,  $q_f$  in F,  $m \geq 0$ ,



$$(q_0, w_1, \dots, w_n, z_0 \uparrow z_R, 0)$$

$$\vdash_M^{a*} (q', w_1, \dots, w_n, z_0 z_1, \dots, z_m z_R \uparrow, 0)$$

$$\vdash_M^a (q_1, w_1, \dots, w_n, z_0 \uparrow z_1, \dots, z_m, 1)$$

$$\vdash_M^{c*} (q'_1, w_2, \dots, w_n, z_0 z_1, \dots, z_m z_R \uparrow, 1)$$

$$\vdash_M^c (q_2, w_2, \dots, w_n, z_0 \uparrow z_1, \dots, z_m, 2)$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$\vdash_M^c (q_n, w_n, z_0 \uparrow z_1, \dots, z_m, n)$$

$$\vdash_M^{c*} (q', \epsilon, z_0 z_1 \dots z_m z_R \uparrow, n)$$

$$\vdash_M^c (q_f, \epsilon, z_0 z_1 \dots z_m z_R \uparrow, n),$$

where  $w = w_1, \dots, w_n$ .

This implies that

$$\left. \begin{aligned} \delta_a(q', z_R) &= \{(q_1, \epsilon, 1)\}, \\ \delta_c(q'_j, \epsilon, z_R, j) &= \{(q_{j+1}, 1)\}, \quad 1 \leq j \leq n-1, \text{ and} \\ \delta_c(q'_n, \epsilon, z_R, n) &= \{(q_f, 0)\}, \end{aligned} \right\} \quad (IV)$$

and that

$$\left. \begin{aligned} (z_0, q_0, q_1, \dots, q_n) \xrightarrow{M} (q', q'_1, \dots, q'_n, \\ w_1, \dots, w_n) \end{aligned} \right\} \quad (V)$$



From the construction of G, (IV) implies that

$\rightarrow ([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n])^n$ , is in P.

Again, in view of A, (V) implies that

$$([q_0, q_1, \dots, q_n, z_0, q', q'_1, \dots, q'_n])^n \\ \xrightarrow[G]{*} (w_1, \dots, w_n).$$

$$\text{Thus, } (S)^n \xrightarrow[G]{*} (w_1, \dots, w_n).$$

Hence the result.

Q.E.D.

It follows from Theorems 3.2.2 and 3.2.1 that

### Theorem 3.2.3:

If M is an nGCA of order k, then we can construct an nGCA M' of order 2 such that M' is in normal form and  $T(M) = T(M')$ .

### 5.3 n-PUSHDOWN ASSEMBLERS:

The definition of n-pushdown assemblers,  $n \geq 1$ , is a generalization of the definition of pushdown assemblers (3) which are pushdown automata with some additional features. We propose these machines as machine models of  $(1, n)$ -coupled translations.

#### Basic Definitions:

Before giving the formal definitions, we shall discuss informally the features and the working of these machines.

Let  $n \geq 1$ ,  $k \geq 1$ . A  $k$ -register,  $n$ -pushdown assembler is a nondeterministic machine which consists of:

a finite control;

an input tape which is read-only, and a 1-way input head;

a pushdown store which is infinite in one direction only, say upwards;

$n$  'waiting' registers which are ordered from left to right and are empty, to start with; and

$n$  sets of  $k$  passive registers each, associated with each symbol<sup>1</sup> of the pushdown store: All the sets of registers and all registers within a set are ordered from left to right. Each register can be empty or hold a string of output symbols (possibly  $\epsilon$ ).

The finite control can read the top symbol of the pushdown store, and in any move can replace the top symbol by a finite length string of symbols (possibly  $\epsilon$ , in which case the top symbol is said to be 'erased'). Let us have the convention that the right-most symbol of the string replaces the top symbol of the pushdown store and the other symbols of the string are stored in order in the upward direction (i.e. the leftmost symbol of the string will become the new top symbol of the pushdown store).

- 
1. A symbol of the pushdown store together with its registers will be called a 'level'.



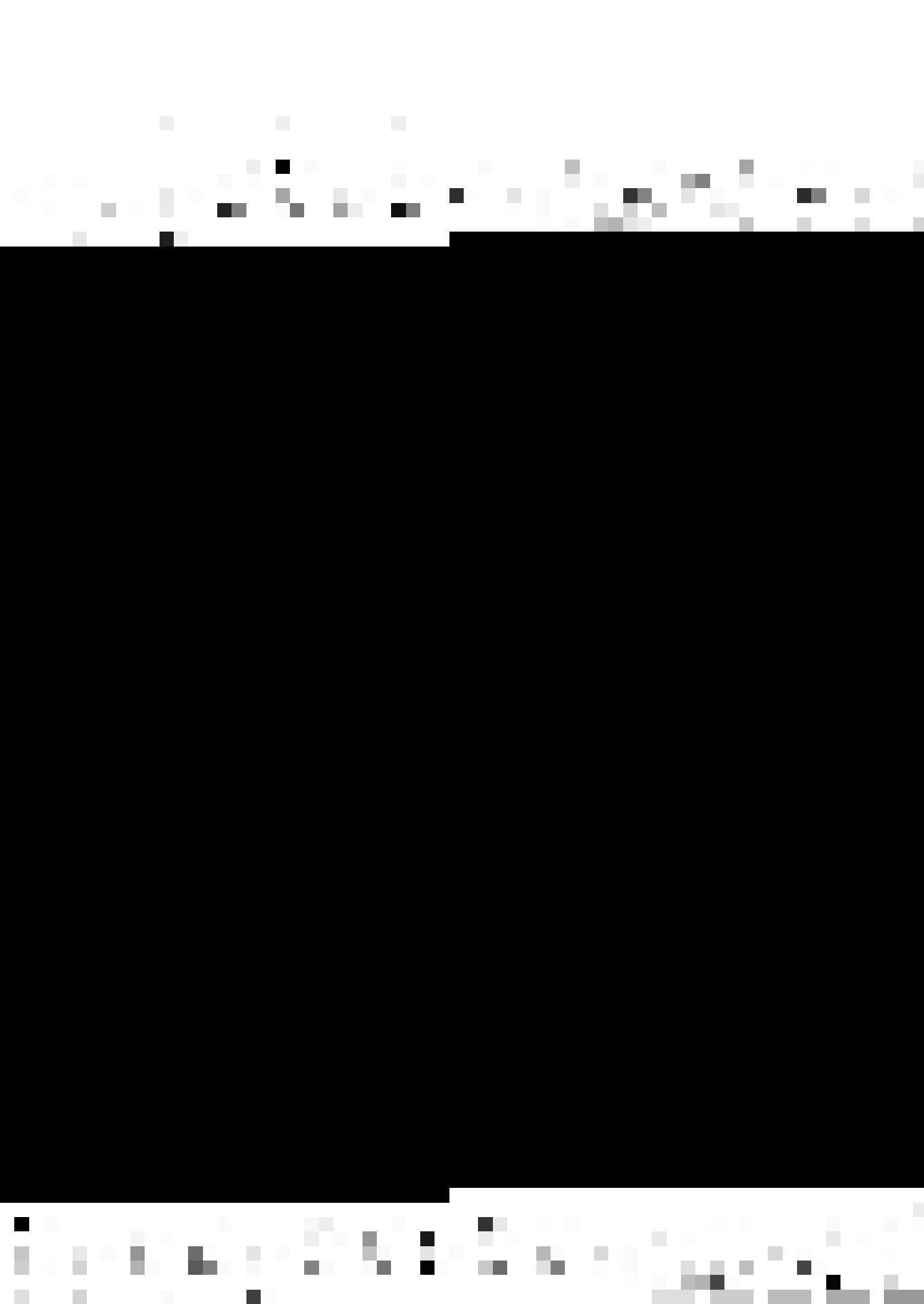
Whenever a (finite) string of length greater than or equal to one replaces the top symbol of the pushdown store, the registers which are associated with the top symbol (which is replaced) get associated with the rightmost symbol of the string, without any change in the ordering of the registers or the contents of the registers.

Again, whenever the top symbol of the pushdown store is erased, the contents of all the  $k$  registers in each set of registers, associated with the symbol erased, are concatenated in order from left to right (if a register is empty, it is treated as though it contained  $\epsilon$ ). The string got by concatenating the contents of the registers in the  $i$ -th set of registers is placed in the  $i$ -th waiting register.

Each move of the machine which erases the top symbol in the pushdown store and is not the last move of the machine is immediately followed by a move which transfers the content of the  $i$ -th waiting register to some (specified) empty register in the  $i$ -th set of registers and then 'empties' all the waiting registers. If the move makes the machine transfer the content of some waiting register to some register which is not empty, the machine gets 'stuck' and cannot make any further move.

The machine can make three types of moves:

One type of move makes the machine replace the top symbol of the pushdown store by a finite string of symbols



(possibly  $\epsilon$ ) with the corresponding creation of additional  $n$  sets of  $k$  registers each, if the replacing string is of length greater than one, as discussed earlier, with a possible change of state.

Another type of move can store  $n$  finite output strings (any of which may possibly be  $\epsilon$ ) - one string in one of the empty registers in each of the  $n$  sets of registers in the top level, in a specified manner.

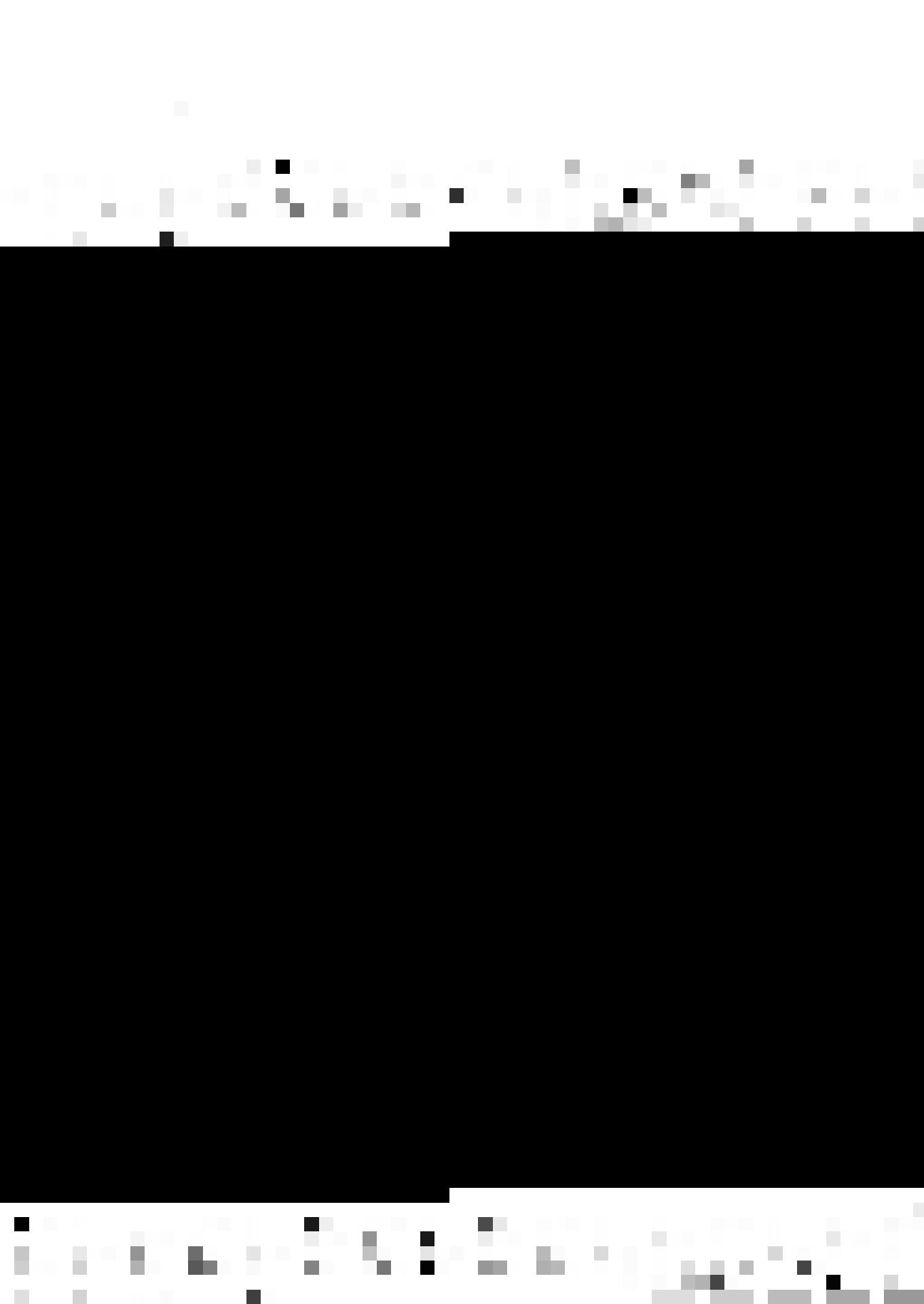
Yet another type of move transfers the contents of the  $i$ -th waiting register,  $1 \leq i \leq n$ , to some empty register in the  $i$ -th set of registers in the top level, in a specified manner, with a possible change in the state of the machine. This type of move immediately follows a move which erases the top symbol of the pushdown store, and cannot take place otherwise.

Formally, a  $k$ -register  $n$ -pushdown assembler ( $n(k)PA$ ) is a 9-tuple:

$$M = (Q, \Sigma, \Delta, P, \lambda, \mu, D, q_0, z_0),$$

where

- (1)  $Q$  is a finite set of states with  $q_0$  as the start state,
- (2)  $\Sigma$  is a finite set of input symbols,
- (3)  $\Delta$  is a finite set of output symbols,
- (4)  $P$  is a finite set of pushdown store symbols, which includes the start symbol  $z_0$ ,



(5)  $\lambda$  is a mapping from  $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$  to finite subsets of  $Q \times \Gamma^*$ .

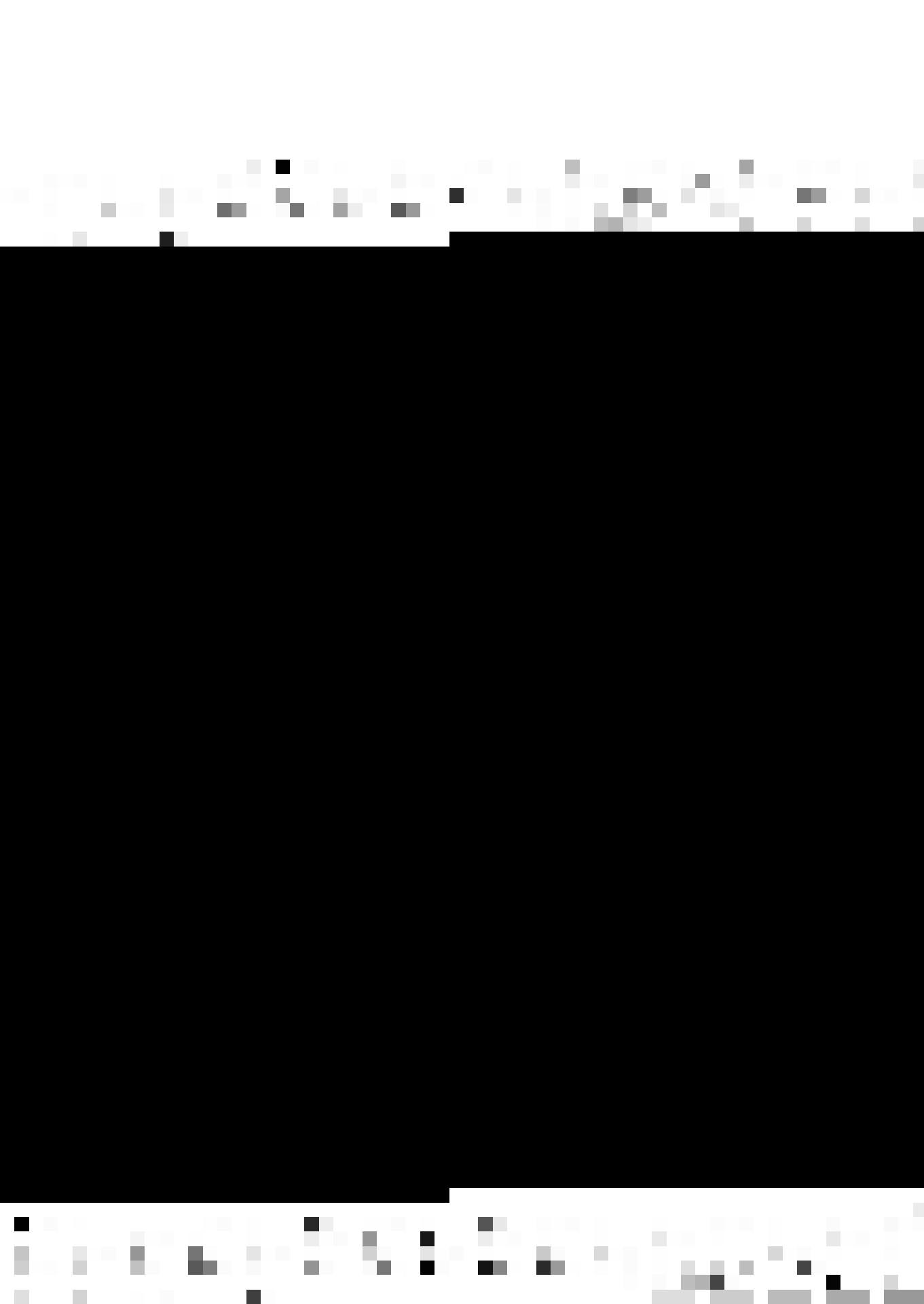
(6)  $\mu$  is a mapping from  $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$  to finite subsets of  $Q \times (\Delta_1^* \times \{1, \dots, k\}) \times \dots \times (\Delta_n^* \times \{1, \dots, k\})$ , where  $\Delta_i, 1 \leq i \leq n, = \Delta$ .

(7)  $\nu$  is a mapping from  $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$  to finite subsets of  $Q \times \{1, \dots, k\}_1 \times \dots \times \{1, \dots, k\}_n$ .

$\lambda$  is the 'next move mapping' which controls changes of pushdown symbols without any change in the register contents.

If  $\lambda(q, a, z)$  contains  $(q', \epsilon)$ , then if the top symbol of the pushdown store is  $Z$ ,  $M$  may erase the top level, change its state from  $q$  to  $q'$ , and use input  $a$  ( $a$  may possibly be  $\epsilon$ ). Also, in this case the contents of the registers in the  $i$ -th set at the top level,  $1 \leq i \leq n$ , will be concatenated, in order, from left to right (with an empty register treated as though it contained  $\epsilon$ ) and the resulting string stored in the  $i$ -th waiting register, provided it is empty. If this move is not the last move of the machine then it must be immediately followed by a move of the machine which transfers the contents of the various waiting registers into various other registers.

If  $\lambda(q, a, z)$  contains  $(q', x_1, \dots, x_m)$ ,  $m \geq 1$ ,  $M$  may replace  $Z$  by  $x_m$  at the top level with everything else remaining the same at this level (i.e. the  $n$  sets of registers with  $k$  1. Here,  $\{1, \dots, k\}_j$  indicates the  $j$ -th set  $\{1, \dots, k\}$ ,  $1 \leq j \leq n$ ).



registers in each set which were associated with the previous pushdown top symbol  $Z$  get now associated with  $X_m$  without any change in the register contents), then grow new levels on the top of the pushdown store with empty registers and pushdown symbols  $X_{m-1}, \dots, X_1$ , in order from below.

$\mu$  is the 'next move mapping' which controls the storing of finite length strings of output symbols in registers.

If  $\mu(q, a, Z)$  contains  $(q', (x_1, i_1), \dots, (x_n, i_n))$ , then when  $Z$  is the top pushdown symbol,  $M$  may use input  $a$ , change its state from  $q$  to  $q'$ , and store output string  $x_j$ ,  $1 \leq j \leq n$ , in the  $i_j$ -th register (in order from the left) of the  $j$ -th set of registers of the top level, provided that register is empty.

$\nu$  is the 'next move mapping' which controls the transferring of the contents of waiting registers into various registers.

If  $\nu(q, a, Z)$  contains  $(q', i_1, \dots, i_n)$ , and if  $Z$  is the top pushdown symbol and the level above has just been erased,  $M$  may use input  $a$ , change the machine from state  $q$  to  $q'$  and transfer the contents of the  $j$ -th waiting register,  $1 \leq j \leq n$ , to the  $i_j$ -th register of the  $j$ -th set of registers of the current top level provided that register is empty and make empty all the waiting registers.



$M$  is an n-pushdown assembler ( $nPA$ ) iff it is an  $n(k)PA$  for some  $k \geq 1$ .

We define a 'configuration' of an  $n(k)PA$   $M = (Q, \Sigma, \Delta, \Gamma, \lambda, \mu, \nu, q_0, z_0)$ , to be a combination of the state of finite control, the nonblank portion of the input tape including and to the right of the cell currently being scanned by the input head, the contents of the pushdown store along with the contents of all the registers associated with each symbol of the pushdown store, and the content of the waiting registers.

Formally, a configuration of  $M$  is a 4-tuple:  $(q, w, \alpha, \beta)$ , where  $q$  is in  $Q$ ,  $w$  is in  $\Sigma^*$ ,  $\alpha$  is a string of the type:  $z_1 t_1 \dots \dots z_m t_m$ ,  $m \geq 1$ , where  $z_1, \dots, z_m$  are in  $\Gamma$  ( $z_1$  is the top pushdown symbol),  $t_i = (t_{i1}, \dots, t_{in})$ ,  $1 \leq i \leq m$ ,  $t_{i1}, \dots, t_{in}$  being ordered  $k$ -tuples of elements in  $\Delta^* \cup \{\emptyset\}$ ,  $\beta$  is either  $(\emptyset)^n$  or an  $n$ -tuple  $(w_1, \dots, w_n)$ , where  $w_i$ ,  $1 \leq i \leq n$ , is in  $\Delta^*$ .  $t_{ij}$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$ , represents the contents of the  $k$  registers in the  $j$ -th set of registers, all in order from the left, associated with  $z_i$ .  $w_j$ ,  $1 \leq j \leq n$ , represents the contents of the  $j$ -th waiting register.  $\emptyset$  denotes an empty register.  $\alpha = \emptyset$ , denotes that the pushdown store is empty.

We say that

$$(q_1, w_1, \alpha_1, \beta_1) \xrightarrow{M} (q_2, w_2, \alpha_2, \beta_2),$$

if configuration  $(q_1, w_1, \alpha_1, \beta_1)$  can become  $(q_2, w_2, \alpha_2, \beta_2)$  by a single move.



If  $(q_1, w_1, \alpha_1, \beta_1)$  can become  $(q_2, w_2, \alpha_2, \beta_2)$  by some number of moves (including zero moves), we write:

$$(q_1, w_1, \alpha_1, \beta_1) \xrightarrow{M^*} (q_2, w_2, \alpha_2, \beta_2).$$

$M$  accepts an input-output ordered pair  $(w, x)$  if  $x$  can be written as  $x_1, \dots, x_n$  and the initial configuration  $(q_0, w, z_0, (t_0)^n, T_0)$ , where  $t_0$  is the  $k$ -tuple  $(\emptyset, \dots, \emptyset)$  and  $T_0$  is the  $n$ -tuple  $(\emptyset, \dots, \emptyset)$ , is such that

$$(q_0, w, z_0(t_0)^n, T_0) \xrightarrow{M^*} (q, \epsilon, \emptyset, (x_1, \dots, x_n)),$$

for any  $q$  in  $Q$ .

We denote by  $T(M)$ , the set of input-output ordered pairs accepted by  $M$ .  $T(M)$  is also called the translation defined by  $M$ .

#### $(1, n, k)$ -coupled translations and $k$ -register $p$ -pushdown assemblers:

We shall now state two theorems which essentially establish that  $(1, n, k)$ -coupled translations,  $k \geq 2$ , are characterized by the translations defined by  $k$ -register  $n$ -pushdown assemblers. The proofs of these theorems are generalizations of the proofs of corresponding theorems which establish that syntax directed translations of order  $k$  are characterized by the translations defined by  $k$ -register pushdown assemblers (3), and are given in Appendix G.



Theorem 3.3.1:

If a translation  $T$  is a  $(1, n, k)$ -coupled translation then  $T = T(M)$  for some  $k'$ -register  $n$ -pushdown assembler  $M$ , where  $k' = k$  if  $k \geq 2$  and  $k' \leq 2$ , otherwise.

Theorem 3.3.2:

If  $T = T(M)$  for a  $k$ -register  $n$ -pushdown assembler  $M$ , then  $T$  is a  $(1, n, k)$ -coupled translation.

3.4 CONCLUSIONS:

In this chapter, we have defined two classes of machines:  $n$ -turn generalized checking automata and  $n$ -pushdown assemblers. The definition of  $n$ -turn generalized checking automata is a generalization of the definition of  $n$ -turn checking automata (35) while the definition of  $n$ -pushdown assemblers is a generalization of pushdown assemblers (3). We have shown that  $n$ -turn generalized checking automata provide a machine characterization of  $n$ -simple coupled languages and  $n$ -pushdown assemblers provide a machine characterization of  $(1, n)$ -coupled translations.

Because the class of simple matrix languages of degree  $n$  is the same as the class of  $n$ -simple coupled languages (see Chapter 5), it follows that we have provided the machine characterization of simple matrix languages of degree  $n$ , as well. Similarly, since the class of 1-simple coupled languages



is the same as the class of context-free languages, it follows that we have provided a new machine characterization of context-free languages.

In view of Theorem 2.2.3(b), it follows that the class of  $n'$ -turn generalized checking automata is a strictly more powerful class of machines than the class of  $n$ -turn checking automata,  $n' > n$ . Again, in view of Theorem 3.3.1 and Theorem 3.3.2, the results of the last chapter (Chapter 2) give the corresponding results for the structures induced on the translations defined by  $k$ -register  $n$ -pushdown assemblers, by the values of  $k$  and  $n$ , and the interrelationships of these structures.

We believe that a minor extension of the features of  $n$ -turn generalized checking automata will result in a class of machines which will characterize the class of  $n$ -coupled languages. Also, it seems that almost all the work concerning pushdown assemblers reported in (3, 6) can be easily generalized to  $n$ -pushdown assemblers.

Nondeterminism is something inherent to the power of  $n$ -turn generalized checking automata. It seems to us, therefore, that deterministic  $n$ -turn generalized checking automata do not comprise an interesting class of machines.

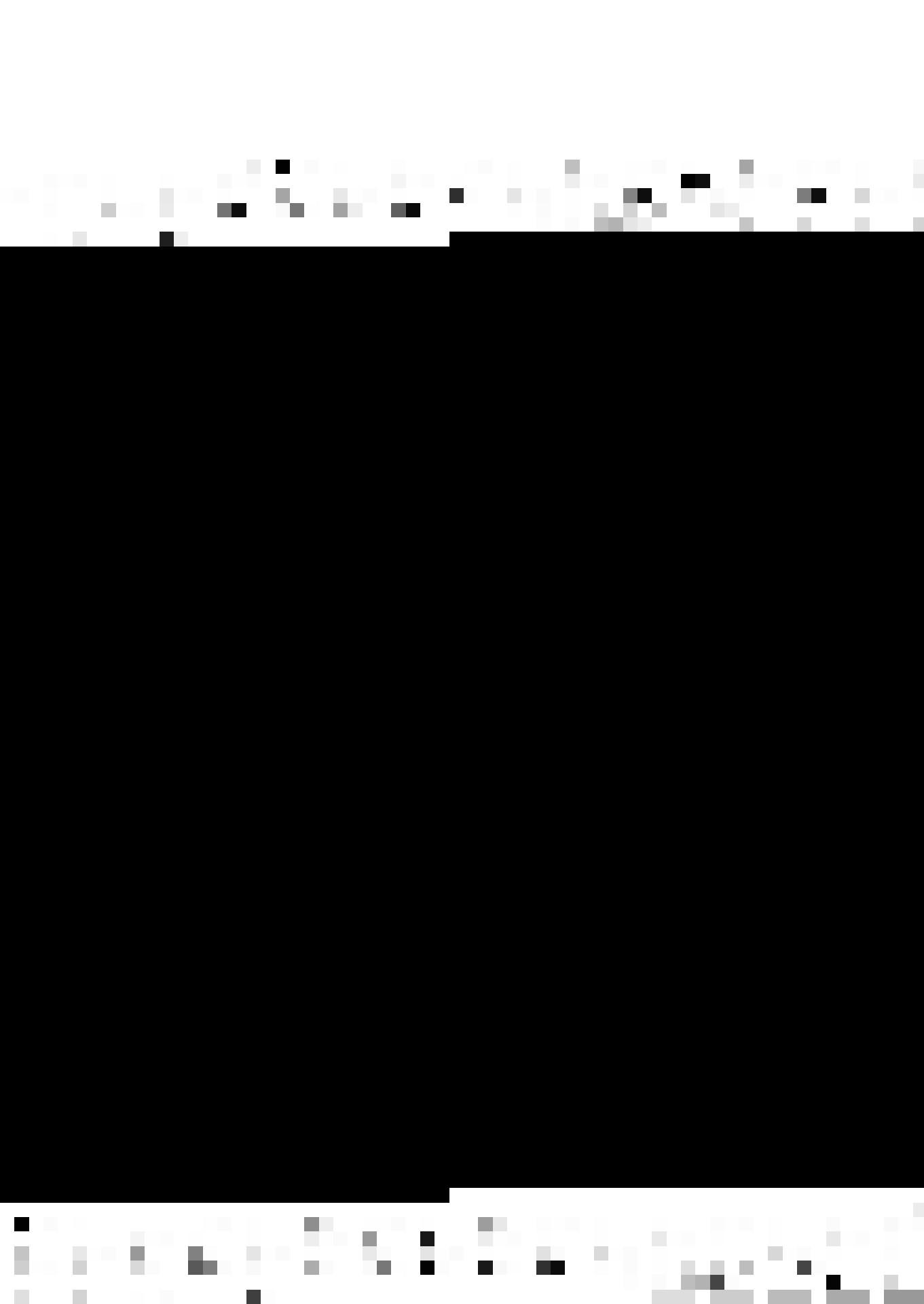
We have not investigated the relationship between the class of  $n$ -turn generalized checking automata and the class



of checking automata (16) i.e. the relationship between the class of n-simple coupled languages and the class of checking automaton languages. This issue is worth looking into.

Machine characterizations of languages and translations have been very useful for developing the theory of languages and translations. We thus believe that the machine models that we have proposed in this chapter will be useful for developing the theory of coupled languages and translations (this fact is supported in Chapter 4 where we use the machine characterization of n-simple coupled languages in order to prove that the class of n-simple coupled languages is closed under intersection with regular sets).

The machine models that we have proposed in this chapter will also be useful for developing systems which are based on n-simple coupled languages or  $(1, n)$ -coupled translations.



## CHAPTER 4

### CLOSURE PROPERTIES AND DECISION PROBLEMS

In this chapter, we investigate some of the closure properties and decision problems of coupled languages. The class of  $n$ -simple coupled languages is the same as the class of simple matrix languages of degree  $n$  (see Chapter 5) whose closure properties and decision problems have been studied in (19). Again, as we show in Chapter 5, the class of  $n$ -right linear coupled languages is the same as the class of equal matrix languages of order  $n$  or the class of right linear simple matrix languages of degree  $n$  whose closure properties and decision problems have been studied in (19,34,35,36). We shall essentially limit our attention to  $n$ -coupled languages and coupled languages.

In section 4.1, we discuss some closure properties. In section 4.2, we discuss some of the decision problems. Finally, we give our conclusions in section 4.3.

#### 4.1 CLOSURE PROPERTIES:

We shall first prove that  $\mathcal{C}_n^L$ ,  $\mathcal{S}_n^L$  as well as  $\mathcal{R}_n^L$  are closed over union,  $\mathcal{C}_n^L$  and  $\mathcal{S}_n^L$  are closed under substitution by context-free languages,  $\mathcal{A}_n^L$  is closed under intersection with regular sets and under sequential transducer



mappings.

Theorem 4.1.1:

$\mathcal{C}_n^L$  (or  $\mathcal{S}_n^L$  or  $\mathcal{R}_n^L$ ) is closed over union.

Proof:

Let  $G_1 = (V_{N_1}, V_{T_1}, P_1, (S_1)^n)$  and  $G_2 = (V_{N_2}, V_{T_2}, P_2, (S_2)^n)$  be two nCG's (respectively, nSCG's) [respectively, nRLCG's]. Without loss of generality, let  $V_{N_1} \cap V_{N_2} = \emptyset$ .

Construct a new grammar  $G_3 = (V_{N_1} \cup V_{N_2} \cup \{S_3\}, V_{T_1} \cup V_{T_2}, P_3, (S_3)^n)$ , where  $S_3 \cap (V_{N_1} \cup V_{N_2}) = \emptyset$  and  $P_3 = \{S_3 \rightarrow (S_1)^n\} \cup \{S_3 \rightarrow (S_2)^n\} \cup P_1 \cup P_2$ .

Clearly,  $G_3$  is an nCG (respectively, nSCG), [respectively, nRLCG] and  $L(G_3) = L(G_1) \cup L(G_2)$ .

Q.E.D.

Theorem 4.1.2:

$\mathcal{C}_n^L$  (or  $\mathcal{S}_n^L$ ) is closed under substitution by context-free languages.

Proof:

Consider an nCG (respectively, nSCG)  $G = (V_N, \{a_1, \dots, a_k\}, P, (S)^n)$  in normal form. Let  $G_i = (V_{N_i}, V_{T_i}, P_i, S_i)$  be a context-free grammar in Chomsky Normal Form (8) generating the set  $f(a_i)$ , for each  $i$ ,  $1 \leq i \leq k$ .



Let  $r_{G_i} = (r_{V_{N_i}}, V_{T_i}, r_{P_i}, (r_{S_i})^n)$ ,  $1 \leq i \leq n$ , be an nSCG constructed from  $G_i$  as follows:

$r_{V_{N_i}} = \{r_A | A \text{ is in } V_{N_i}\}$ , and  
 $r_{P_i} = \{r_A \rightarrow (r_{B_1} r_{B_2})^n | A \rightarrow B_1 B_2 \text{ is in } P_i \text{ and } B_1, B_2 \text{ are in } V_{N_i}\} \cup \{r_A \rightarrow (\epsilon, \dots, a, \dots, \epsilon) | A \rightarrow a \text{ is in } P_i\}$ , where the right hand side of the production is an n-tuple with the r-th element as a and all other elements as  $\epsilon | A \rightarrow a$  is in  $P_i$ , a is in  $V_{T_i}\}$ .

We assume without loss of generality that all nonterminal vocabularies are pairwise disjoint.

Construct a new nCG (respectively, nSCG)

$$G' = (V'_N, V'_T, P', (S)^n), \text{ where}$$

$$V'_N = \bigcup_{\substack{1 \leq i \leq k \\ 1 \leq r \leq n}} r_{V_{N_i}} \cup V_N, \text{ and}$$

$$P' = \bigcup_{\substack{1 \leq i \leq k \\ 1 \leq r \leq n}} r_{P_i} \cup \left\{ A \rightarrow \alpha | A \rightarrow \alpha \text{ is a nonterminating production in } P \right\} \cup \left\{ A \rightarrow ({}^1 S_{i_1} \dots {}^n S_{i_n})^n | A \rightarrow (a_{i_1}, \dots, a_{i_n}) \text{ is a terminating production in } P \text{ and } {}^j S_{i_j} = \epsilon \text{ iff } a_{i_j} = \epsilon \right\}.$$

It is easily seen that  $f(L(G)) = L(G')$ .

Q.E.D.

Cor. 4.1.2.1:

$C_n^L$  (or  $\mathcal{S}_n^L$ ) is closed under substitution by regular sets.



Cor. 4.1.2.2:

$\mathcal{C}_n^L$  (or  $\mathcal{S}_n^L$ ) is closed under homomorphism.

Theorem 4.1.3:

If  $L$  is in  $\mathcal{S}_n^L$  and  $R$  is a regular set, then  $L \cap R$  is in  $\mathcal{S}_n^L$ .

Proof:

Let  $M = (Q_M, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_0, z_R, F_M)$ , be an nGCA accepting  $L$  and  $A = (Q_A, \Sigma, \delta, p_0, F_A)$ , a deterministic finite automaton accepting  $R$ . The nGCA

$M' = (Q_M \times Q_A, \Sigma, \delta'_a, \delta'_c, C, [q_0, p_0], z_0, z_R, F_M \times F_A)$ , where  $\delta'_a$  and  $\delta'_c$  are defined below, accepts  $L \cap R$ .

For all  $q$  in  $Q_M$ ,  $p$  in  $Q_A$ ,  $a$  in  $\Sigma \cup \{\epsilon\}$ ,  $z$  in  $\Gamma$ , and  $j$ ,  $1 \leq j \leq n$ ,  $\delta'_a([q, p], z)$  contains  $([q', p], \alpha, b)$  whenever  $\delta_a(q, z)$  contains  $(q', \alpha, b)$ , and  $\delta'_c([q, p], a, z, j)$  contains  $([q', \delta(p, a)], b)$  whenever  $\delta_c(q, a, z, j)$  contains  $(q', b)$ . ( $\delta(p, \epsilon)$  is  $p$  for all  $p$  in  $Q_A$ ).

Informally,  $M'$  keeps track of the states of  $M$  and  $A$  in its finite control.

It is easy to show that  $T(M) = L \cap R$ ; the proof is on the same lines as that of the corresponding result for context-free languages (18), and is omitted.

Q.E.D.



Cor. 4.1.3.1:

If  $L$  is in  $\mathcal{B}_n^L$  and  $R$  is a regular set, then  $L - R$  is in  $\mathcal{B}_n^L$ .

\*\*\*

Note that if  $R$  is over the alphabet  $\Sigma$ , then  $L - R = L \cap (\Sigma^* - R)$  and the class of regular sets is closed under complementation.

\*\*\*

Ginsburg(14) proves that an arbitrary class of languages which contains all regular sets and which is closed under the operations of union, substitution by regular sets, and intersection with regular sets, is closed under sequential transducer mappings.

Thus, we have:

Theorem 4.1.4:

$\mathcal{B}_n^L$  is closed under sequential transducer mappings.

\*\*\*

Now we prove that  $\mathcal{C}_n^L$  as well as  $\mathcal{B}_n^L$  are not closed under intersection or complementation.

Theorem 4.1.5:

$\mathcal{C}_n^L$  (or  $\mathcal{B}_n^L$ ) is not closed under intersection or complementation.



Proof:

(a) Intersection:

The languages:

$$L_1 = \left\{ a_1^i a_2^i \dots a_{2n}^i a_{2n+1}^p \mid i \geq 1, p \geq 1 \right\}, \text{ and}$$

$$L_2 = \left\{ a_1^q a_2^i a_3^i \dots a_{2n+1}^i \mid i \geq 1, q \geq 1 \right\},$$

are nSCI's since they are generated by the nSCG's  $G_1$  and  $G_2$ , respectively:

$$G_1 = (\{S, A\}, \{a_1, \dots, a_{2n+1}\}, P, (S)^n),$$

where P comprises of the productions:

$$S \rightarrow (S, \dots, S, Sa_{2n+1}),$$

$$S \rightarrow (a_1 A a_2, a_3 A a_4, \dots, a_{2n-1} A a_{2n}),$$

$$S \rightarrow (a_1 a_2, a_3 a_4, \dots, a_{2n-1} a_{2n}), \text{ and}$$

$$A \rightarrow (a_1 a_2, a_3 a_4, \dots, a_{2n-1} a_{2n}).$$

Again,

$$G_2 = (\{S, A\}, \{a_1, \dots, a_{2n+1}\}, P, (S)^n),$$

where P comprises of the productions:

$$S \rightarrow (Sa_4, S, \dots, S),$$

$$S \rightarrow (a_2 A a_3, a_4 A a_5, \dots, a_{2n} A a_{2n+1}),$$

$$S \rightarrow (a_2 a_3, a_4 a_5, \dots, a_{2n} a_{2n+1}), \text{ and}$$

$$A \rightarrow (a_2 a_3, a_4 a_5, \dots, a_{2n} a_{2n+1}).$$

Now,  $L_1 \cap L_2 = \left\{ a_1^i a_2^i \dots a_{2n+1}^i \mid i \geq 1 \right\}$ , which is not an nCL (see the proof of Lemma 2.2.1) and hence is not an nSCL.



(b) Complementation:

Since  $\mathcal{C}_n^L$  (or  $\mathcal{S}_n^L$ ) is closed over union but not under intersection, it cannot be closed under complementation since  $L_1 \cap L_2 = \overline{\overline{L}_1 \cup \overline{L}_2}$ .

Q.E.D.

Before we proceed further, we will prove the following lemmas which will be useful later.

Lemma 4.1.1:

The language:

$$L = \left\{ a_1^{m_1} b_1^{m_1} c_1^{m_1} a_2^{m_2} b_2^{m_2} c_2^{m_2} \dots a_n^{m_n} b_n^{m_n} c_n^{m_n} \mid m_i \geq 1, \right. \\ \left. 1 \leq i \leq n \right\},$$

is not an nCL.

Proof:

Let us suppose for the time being that L is an nCL and is generated by an nCG G in normal form.

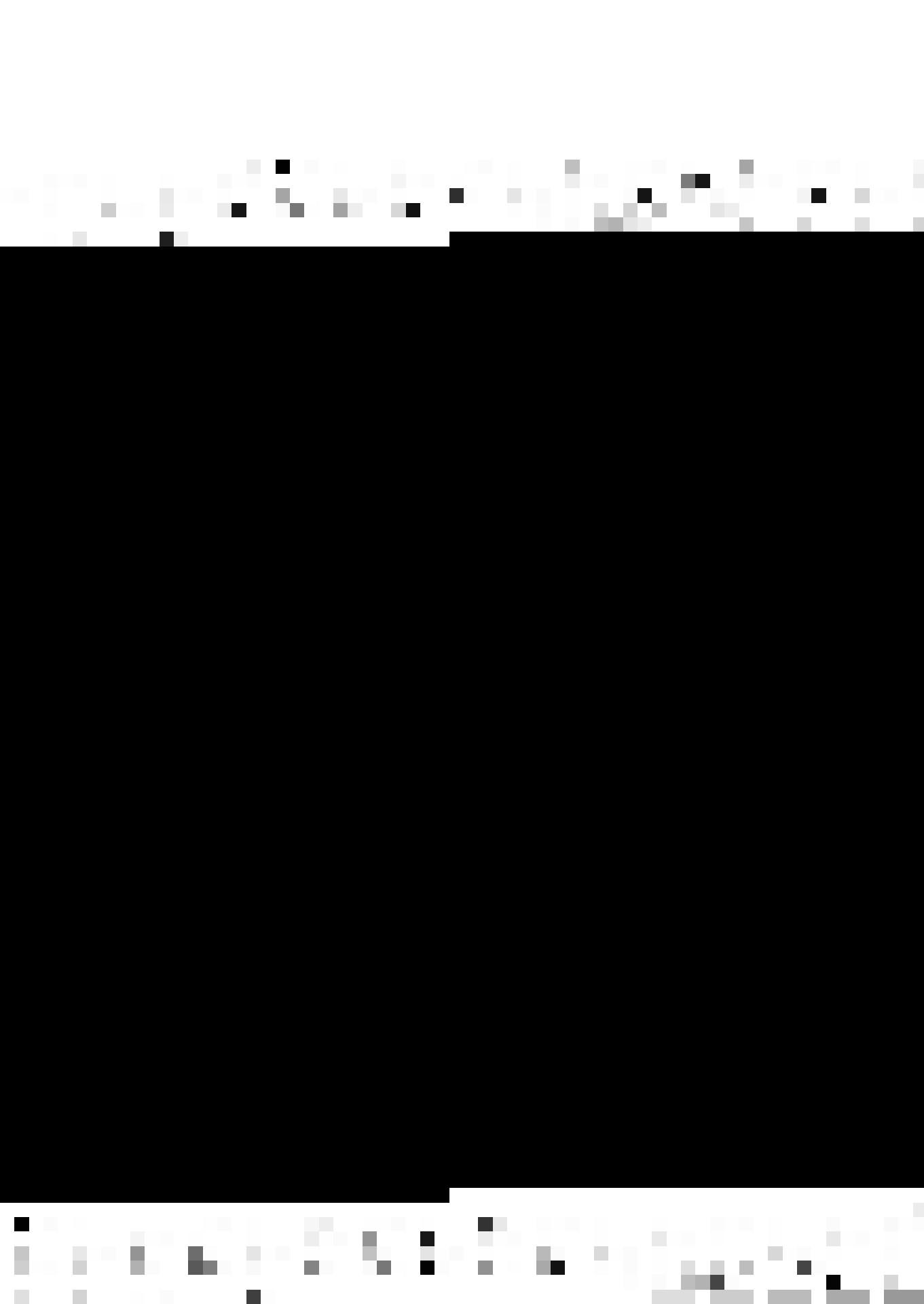
Consider the sentence

$$z = a_1^n \cdot p b_1^n \cdot p c_1^n \cdot p \dots a_n^n \cdot p b_n^n \cdot p c_n^n \cdot p,$$

where p is the constant referred to in Theorem 2.1.1.

Let  $z_i = a_i^n \cdot p b_i^n \cdot p c_i^n \cdot p$ , for some  $i$ ,  $1 \leq i \leq n$ . Then, according to Theorem 2.1.1, z can be represented as

$u_1 v_1 w_1 x_1 y_1 \dots u_n v_n w_n x_n y_n$ , where  $b(i)$  and  $b(ii)$  of the theorem (Theorem 2.1.1) are true, such that  $u_1 v_1^s w_1^s x_1^s y_1 \dots u_n v_n^s w_n^s x_n^s y_n$ ,  $s \geq 0$ , is also in L, and for some j,



$v_j$  or  $x_j$  contains the occurrence of some symbol in  
 $a_i^{n \cdot p} b_i^{n \cdot p} c_i^{n \cdot p}$ .

It can be shown (for a similar proof, see the proof of Lemma 2.2.1) that:

1. None of the  $v_1, x_1, \dots, v_n, x_n$ , can have occurrences of more than one distinct symbol.

2. It cannot be that none of  $v_1, x_1, \dots, v_n, x_n$  contain any occurrences of  $a_i$  or that of  $b_i$  or that of  $c_i$ .

Now, let  $z_1 = a_1^{n \cdot p} b_1^{n \cdot p} c_1^{n \cdot p}$ . From 1 and 2 above, it follows that  $c_1$  occurs in the result of the  $t$ -th tree of the  $n$ -derivation tree of  $z$  in  $G$ ,  $t \geq 2$ .

Again, let  $z_1 = a_2^{n \cdot p} b_2^{n \cdot p} c_2^{n \cdot p}$ . Repeating the same argument,  $a_2$  cannot be in the result of the first tree of the  $n$ -derivation tree of  $z$  in  $G$ , and  $c_2$  is in the result of the  $t$ -th tree,  $t \geq 3$ .

Similarly,  $c_n$  cannot be in the result of the  $n$ -th tree of the  $n$ -derivation tree of  $z$  in  $G$  — a contradiction, because there are just  $n$  trees in the  $n$ -derivation tree of  $z$ .

Hence  $L$  is not an  $n$ CL.

Q.E.D.



Lemma 4.1.2:

The language;

$$L = \left\{ a_1^{m_1} b_1^{m_1} c_1^{m_1} \dots a_n^{m_n} b_n^{m_n} c_n^{m_n} \mid m_i, 1 \leq i \leq n, \geq 1 \right\},$$

is not an nCL.

Proof:

Similar to that of Lemma 4.1.1.

\*\*\*

We shall now show that while  $\mathcal{C}_n^L$  (respectively,  $\mathcal{S}_n^L$ ) is closed under concatenation with  $\mathcal{C}_1^L$  ( $\mathcal{C}_1^L$  is the same as  $\mathcal{S}_1^L$  or the class of context-free languages), it is not closed under concatenation with  $\mathcal{C}_2^L$  (respectively,  $\mathcal{S}_2^L$ ). On the other hand, we shall show that  $\mathcal{C}^L$  (or  $\mathcal{S}^L$ ) is closed under concatenation.

Theorem 4.1.6:

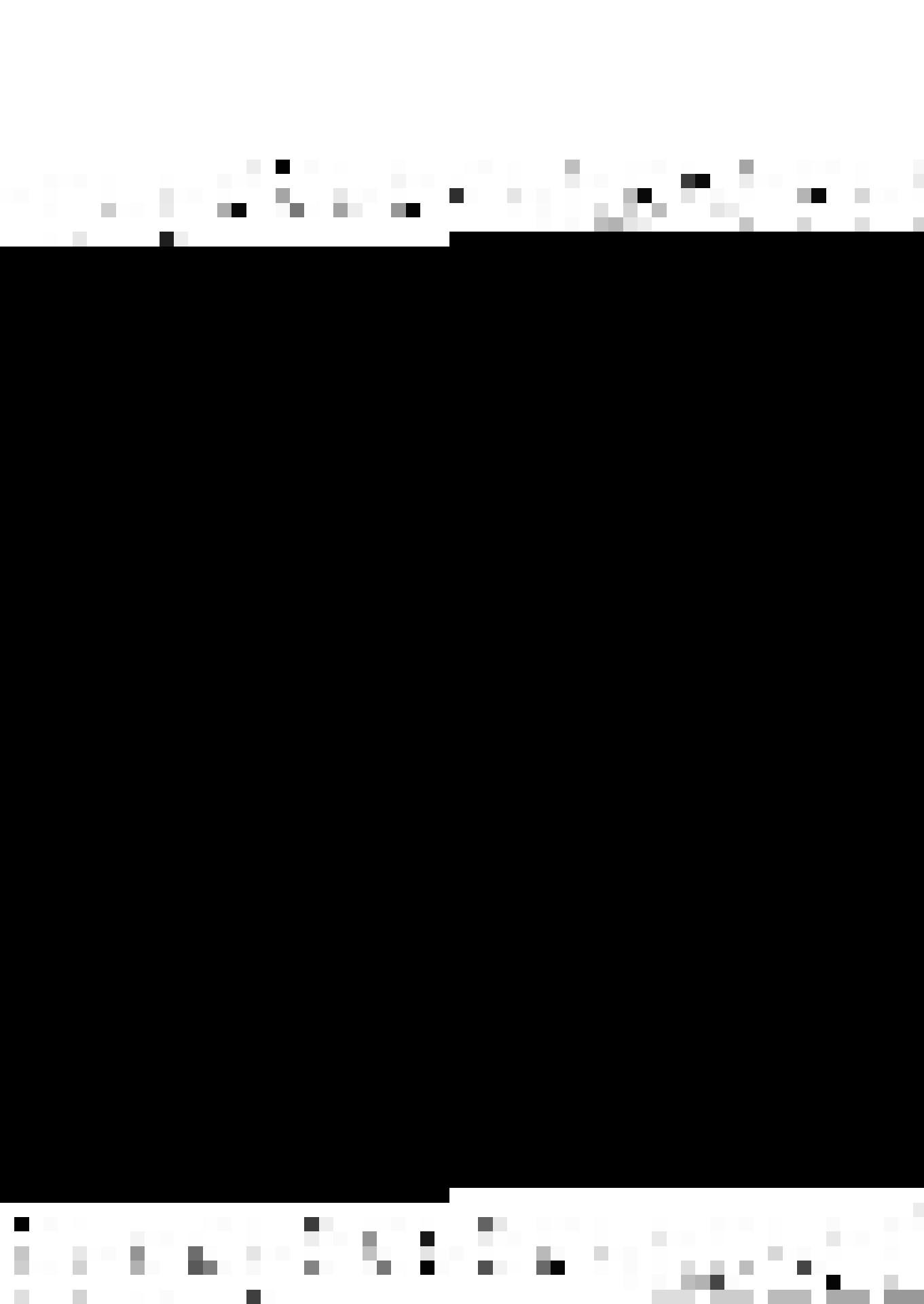
(a)  $\mathcal{C}_n^L$  (or  $\mathcal{S}_n^L$ ) is closed under concatenation with context-free languages.

(b)  $\mathcal{C}_n^L$  (respectively,  $\mathcal{S}_n^L$ ) is not closed under concatenation with languages in  $\mathcal{C}_2^L$  (respectively,  $\mathcal{S}_2^L$ ).

(c)  $\mathcal{C}^L$  (or  $\mathcal{S}^L$ ) is closed under concatenation.

Proof:

(a) Let  $G = (V_N, V_T, P, (S)^n)$  be the given nCG (respectively, nSCG). Let  $G_1 = (V_{N_1}, V_{T_1}, P_1, S_1)$  be the given context-free grammar in Chomsky Normal Form such that



$V_N \cap V_{N_1} = \emptyset$ . From  $G_1$ , construct an nSCG  $G_2$  as follows:

$G_2 = (V_{N_1}, V_{T_1}, P_1, (S_1)^n)$ , where for each production:  $A \rightarrow BC$  in  $P_1$ ,  $B, C$  are in  $V_{N_1}$ , we have:  $A \rightarrow (BC)^n$  in  $P'_1$ , and for each production:  $A \rightarrow a$  in  $P_1$ , where  $a$  is in  $V_T \cup \{\epsilon\}$ , we have:  $A \rightarrow (\epsilon, \dots, \epsilon, a)$ , in  $P'_1$ .

Now, construct

$$G_3 = (V_N \cup V_{N_1} \cup \{S_3\}, V_T \cup V_{T_1}, P_3, (S_3)^n),$$

where  $S_3$  is a symbol which is not in  $V_N \cup V_{N_1}$ , as follows:

$P_3$  comprises of the production:  $S_3 \rightarrow (SS_1)^n$ , and all productions in  $P$  and  $P'_1$ .

It is clear that  $L(G_3) = L(G)L(G_1)$ .

Hence the result.

(b) Let  $L_1 = \{\epsilon\}$  and  $L_2 = \{a_1^i a_2^i a_3^i \mid i \geq 1\}$ . It is easy to show that  $L_1$  is in  $\mathcal{L}_n^L$  and  $L_2$  is in  $\mathcal{L}_2^L$ . It then follows that  $L_1$  is also in  $\mathcal{C}_n^L$  and  $L_2$  is also in  $\mathcal{C}_2^L$ . If  $\mathcal{C}_n^L$  (respectively,  $\mathcal{L}_n^L$ ) were closed under concatenation with languages in  $\mathcal{C}_2^L$  (respectively,  $\mathcal{L}_2^L$ ), then<sup>1</sup>

$$L_1 L_2^n = \{a_1^{m_1} a_2^{m_2} a_3^{m_3} \dots a_1^{m_n} a_2^{m_n} a_3^{m_n} \mid m_i, 1 \leq i \leq n, \geq 1\},$$

would be in  $\mathcal{C}_n^L$ . But this is a contradiction to Lemma 4.1.2.

Hence the result.

1. If  $X$  is a set then  $X^i$  means concatenation of  $X$  with itself  $i$  times.



(c) Let  $G_1 = (V_{N_1}, V_{T_1}, P_1, (S_1)^m)$ , be an mCG (respectively, mSCG), and  $G_2 = (V_{N_2}, V_{T_2}, P_2, (S_2)^n)$ , be an nCG (respectively, nSCG), such that  $L(G_1)$  is the given language in  $\mathcal{C}_m^L$  (respectively,  $\mathcal{S}_m^L$ ) and  $L(G_2)$  is the given language in  $\mathcal{C}_n^L$  (respectively,  $\mathcal{S}_n^L$ ). Without loss of any generality, we can assume that each of  $G_1$  and  $G_2$  is in normal form and  $V_{N_1} \cap V_{N_2} = \emptyset$ .

In order to prove the result, we shall construct an  $(m+n-1)$ CG (respectively,  $(m+n-1)$ SCG)  $G$  such that  $L(G) = L(G_1)L(G_2)$ .

$$G = (V_{N_1} \cup V_{N_2}, V_{T_1} \cup V_{T_2}, P, (S)^{m+n-1}),$$

where  $S \cap V_{N_1} \cap V_{N_2} = \emptyset$  and  $P$  comprises of the following productions:

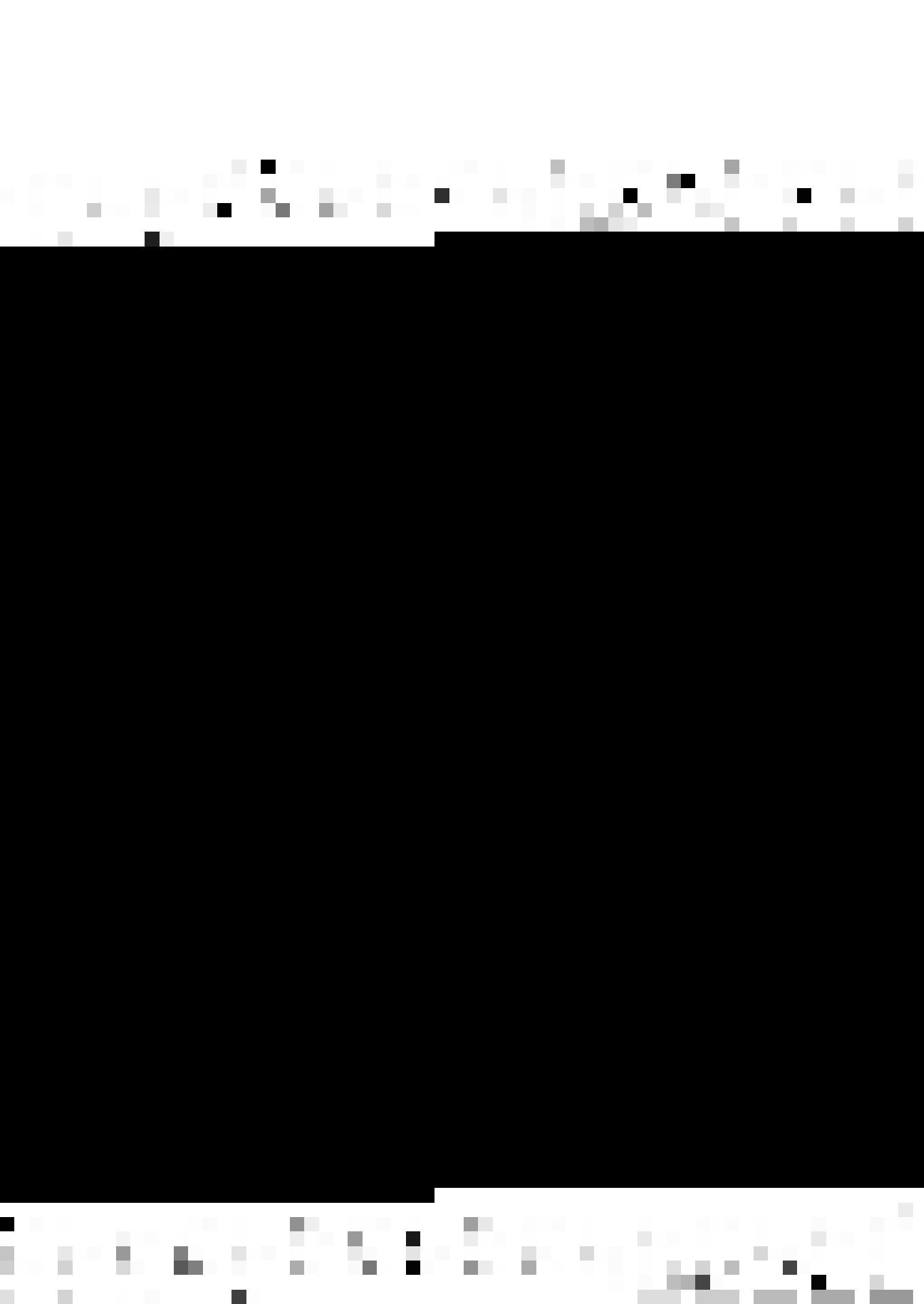
$$1. \quad S \rightarrow (S_1 S_2)^{m+n-1}$$

2. For each nonterminating production:  $A \rightarrow (\alpha_1, \dots, \alpha_m)$ , in  $P_1$ , we have:  $A \rightarrow (\alpha_1, \dots, \alpha_m, \alpha_1, \dots, \alpha_1)$ , in  $P$ .

3. For each terminating production:  $A \rightarrow (a_1, \dots, a_m, \epsilon, \dots, \epsilon)$ , in  $P_1$ , we have:  $A \rightarrow (a_1, \dots, a_m, \epsilon, \dots, \epsilon)$ , in  $P$ .

4. For each nonterminating production:  $A \rightarrow (\alpha_1, \dots, \alpha_n)$ , in  $P_2$ , we have:  $A \rightarrow (\alpha_n, \dots, \alpha_n, \alpha_1, \dots, \alpha_n)$ , in  $P$ .

5. For each terminating production:  $A \rightarrow (a_1, \dots, a_n)$ , in  $P_2$ , we have:  $A \rightarrow (\epsilon, \dots, \epsilon, a_1, \dots, a_n)$ , in  $P$ .



It is clear that  $L(G) = L(G_1)L(G_2)$ .

Hence the result.

Q.E.D.

Given a coupled language, there can be various coupled grammars of different degrees which generate this language. In fact if an nCG  $G$  generates a language  $L$ , we can construct an mCG,  $m \geq n$  which generates the same language. Thus it would be useful and interesting to define a concept about coupled languages which reveals some of their inherent properties. The same holds true for simple coupled languages as well. We define this concept through the following definition:

The rank of a coupled language (respectively, simple coupled language)  $L$  is defined to be the smallest integer  $m$  such that  $L$  is generated by an mCG (respectively, mSCG).

\*\*\*

From the proof of Theorem 4.1.6(c), we can say that if  $L_1$  and  $L_2$  are coupled languages (or simple coupled languages) of rank  $m_1$  and  $m_2$ , respectively, then the rank of the language  $L = L_1L_2$  is less than or equal to  $m_1 + m_2 - 1$ . But is this a tight bound? In the following theorem, we show that it is in fact so.



Theorem 4.1.7:

If  $L_1$  and  $L_2$  are coupled languages (or simple coupled languages) of rank  $m_1$  and  $m_2$ , respectively, then the rank of the language  $L = L_1 L_2$  is less than or equal to  $m_1 + m_2 - 1$ . Moreover,  $m_1 + m_2 - 1$  is a tight bound.

Proof:

In view of the proof of Theorem 4.1.6(c), it suffices to prove only the latter part of the theorem viz. that  $m_1 + m_2 - 1$  is a tight bound.

Consider the languages:

$$L_1 = \left\{ a_1^k \dots a_{2m_1}^k \mid k \geq 1 \right\}, \text{ and}$$

$$L_2 = \left\{ b_1^l \dots b_{2m_2}^l \mid l \geq 1 \right\}.$$

Clearly,  $L_1$  is an  $m_1$  SCL and hence also an  $m_1$  CL. Similarly,  $L_2$  is an  $m_2$  SCL and hence also an  $m_2$  CL. Moreover, it follows from the proof of Lemma 2.2.1 that the rank of  $L_1$  is  $m_1$  and that of  $L_2$  is  $m_2$ . We shall now prove that  $L = L_1 L_2$  is not an  $(m_1 + m_2 - 2)$  CL (and hence cannot be an  $(m_1 + m_2 - 2)$  SCL). In order to prove this, we shall use Theorem 2.1.1.

Suppose for the time being that  $L$  is an  $(m_1 + m_2 - 2)$  CL and is generated by an  $(m_1 + m_2 - 2)$  CG in normal form.

Consider the sentence

$$z = a_1^t \dots a_{2m_1}^t b_1^t \dots b_{2m_2}^t, \text{ in } L,$$



where  $t = (m_1 + m_2 - 2) \cdot p$  and  $p$  is the constant referred to in Theorem 2.1.1. Let  $z_1 = a_1^t \dots a_{2m_1}^t$ . Then, according to Theorem 2.1.1,  $z$  can be represented as:

$$z = u_1 v_1 w_1 x_1 y_1 \dots u_s v_s w_s x_s y_s,$$

where  $s = m_1 + m_2 - 2$ , such that  $u_1 v_1^i w_1 x_1^i y_1 \dots u_s v_s^i w_s x_s^i y_s$  is also in  $L$  for all  $i \geq 0$ , and for some  $j$ ,  $1 \leq j \leq s$ ,  $v_j$  or  $x_j$  contains the occurrence of some symbol in  $z_1$ . From the nature of the language it follows (using similar arguments as used in the proof of Lemma 2.2.1) that:

1. Each of the symbols  $a_1, \dots, a_{2m}$  is contained in  $v_i$  or  $x_i$ .
2. None of  $v_1, x_1, \dots, v_s, x_s$  contains the occurrence of two or more different symbols.

It, therefore, follows that  $b_1$  is contained in the result of the  $r$ -th tree,  $r \geq m_1$ , of the  $s$ -derivation tree of  $z$  in  $G$ .

Again, let  $z_1 = b_1^t \dots b_{2m_2}^t$ . Then, using Theorem 2.1.1, it can be shown in a manner similar to above that  $b_1$  is contained in the result of the  $r$ -th tree,  $r \leq m_1 + m_2 - 2 - m_2 + 1 = m_1 - 1$ , of an  $s$ -derivation tree of  $z$  in  $G$ .

This leads to a contradiction. Hence the result.

Q.E.D.

4.2 DECISION PROBLEMS:

In this section, we shall prove the results for some of the decision problems of  $C_n^L$ , which are decidable. These results obviously hold for  $\lambda_n^L$  as well. As far as the results for the various decision problems for  $C_n^L$  and  $\lambda_n^L$  are concerned, which are undecidable, we simply list those which follow from the corresponding results for context-free languages.

We shall first consider the results of the decidable decision problems of  $C_n^L$ . We shall prove that the nCG's are recursive, and that it is recursively solvable to determine whether for an nCG  $G$ ,  $L(G)$  is empty, finite, or infinite.

In order to prove the recursiveness result (Theorem 4.2.1), we shall first prove a lemma.

Lemma 4.2.1:

Let  $G = (V_N, V_T, P, (S)^n)$  be an nCG such that there is no production of the type:  $A \rightarrow (\epsilon)^n$ , in  $G$ . Let  $((\beta_1, \pi_1), \dots, (\beta_n, \pi_n))$  be a form in  $G$  such that  $|\beta_1 \dots \beta_n| > n \cdot k$ ,  $k \geq 1$ . Then,  $((\beta_1, \pi_1), \dots, (\beta_n, \pi_n))$  cannot derive  $(w_1, \dots, w_n)$ , where  $w_i$ ,  $1 \leq i \leq n$ , is in  $V_T^*$ , such that  $|w_1 \dots w_n| \leq k$ .

Proof:

Let  $|\beta_1 \dots \beta_n| = t (> n \cdot k)$ .



There can be two possibilities:

(i) The number of occurrences of nonterminals in each  $\beta_j$ ,  $1 \leq j \leq n$ , is less than or equal to  $k$ .

(ii) The number of occurrences of nonterminals in each  $\beta_j$ ,  $1 \leq j \leq n$ , is greater than  $k$ .

If (i) is the case, let the number of nonterminals in each  $\beta_j$  be  $s$  ( $\leq k$ ), so that the number of occurrences of terminals in  $\beta_1 \dots \beta_n$  is  $t - n.s$  (which is greater than zero).

Clearly, because there is no production of the type:  $A \rightarrow (\epsilon)^n$ , in  $G$ , the reduction of each nonterminal in  $\beta_1$  and the corresponding nonterminals in  $\beta_j$ ,  $2 \leq j \leq n$ , will contribute atleast one symbol to  $w_1 \dots w_n$ . This implies that:

$$\begin{aligned} |w_1 \dots w_n| &\geq t - n.s + s = t - s.(n-1) \geq t - k.(n-1) \\ &\quad (\text{since } k \geq s) \\ &= t - n.k + k > k \quad (\text{because } t > n.k). \end{aligned}$$

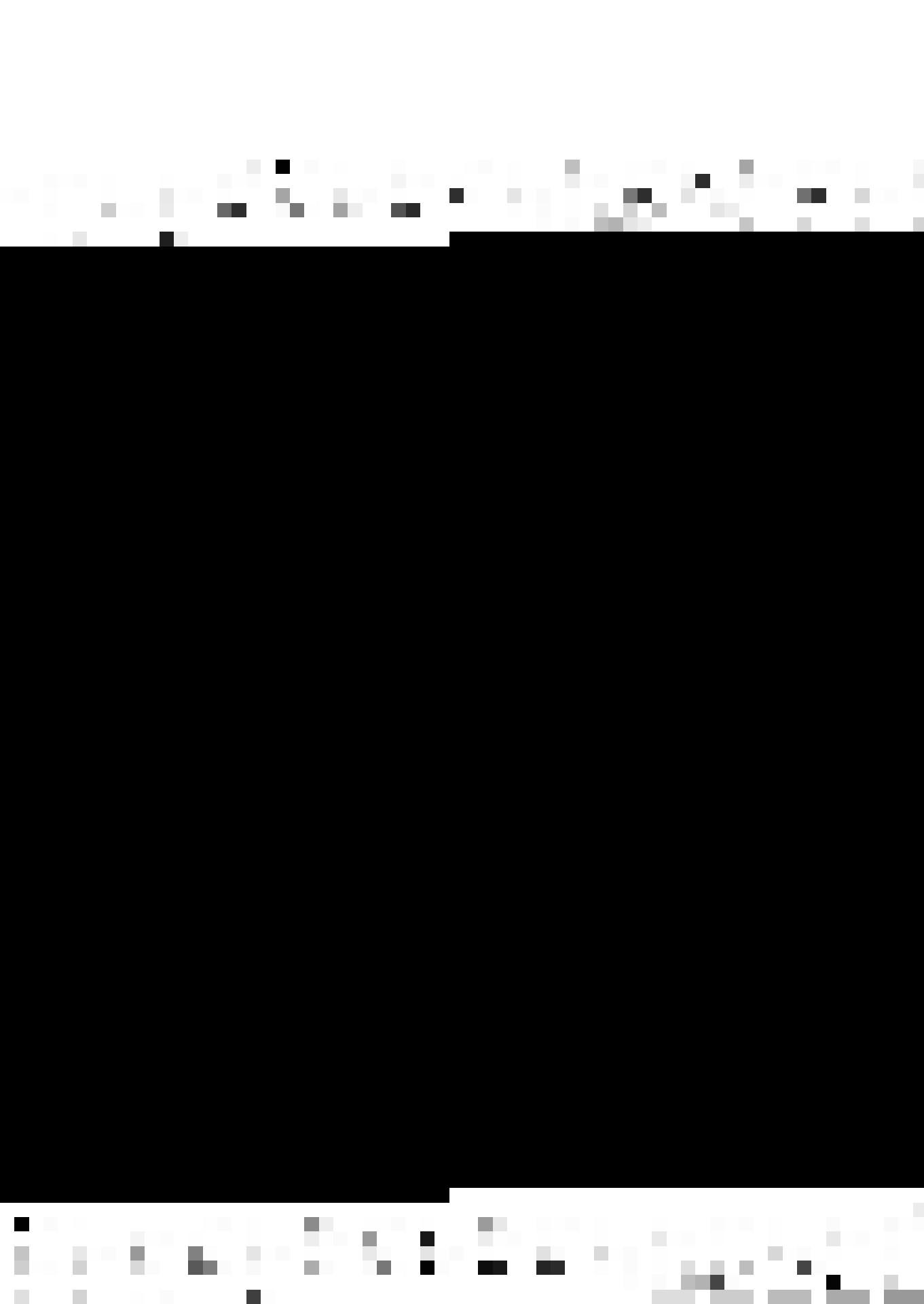
If (ii) is the case, the result is immediate.

Hence the result.

Q.E.D.

#### Theorem 4.2.1:

If  $G = (V_N, V_T, P, (S)^n)$  is an nCG, then  $G$  is recursive.



Proof:

Without loss of any generality, let us assume that  $G$  is in normal form. Obviously,  $\epsilon$  is in  $L(G)$  iff  $P$  contains the production:  $S \rightarrow (\epsilon)^n$ . Let us remove this production from  $P$  (if it is in  $P$ ) so as to form  $P'$ . Clearly,  $G' = (V_N, V_T, P', (S)^n)$ , is such that  $L(G') = L(G) - \{\epsilon\}$ .

Let  $w$  be in  $V_T^*$ . Suppose that  $|w| = k$ . Now we give an algorithm for finding whether  $w$  is in  $L(G)$ .

Algorithm:

1. Let  $T_0 = \{(S)^n\}$ . Set  $i = 1$ .
2. Let  $T_i = T_{i-1} \cup \left\{ \beta = ((\beta_1, \pi_1), \dots, (\beta_n, \pi_n)) \mid \text{for some } \alpha = ((\alpha_1, \pi'_1), \dots, (\alpha_n, \pi'_n)) \text{ in } T_{i-1}, \alpha \xrightarrow[G']{} \beta, \text{ and } |\beta_1 \dots \beta_n| \leq n \cdot k \right\}$ .
3. If  $T_i \neq T_{i-1}$ , then set  $i = i + 1$  and go to step 2. Otherwise, let  $T_e = T_i$ .
4. If there is no  $(w_1, \dots, w_n)$  in  $T_e$  such that  $w = w_1 \dots w_n$ , then  $w$  is not in  $L(G')$ . Otherwise  $w$  is in  $L(G')$ .

Observe that in view of Lemma 4.2.1, it cannot be that  $(S)^n \xrightarrow[G']{} ((\beta_1, \pi_1), \dots, (\beta_n, \pi_n)) \xrightarrow[G']{} (w_1, \dots, w_n)$ , where  $w_i$ ,  $1 \leq i \leq n$ , is in  $V_T^*$ , such  $|w_1 \dots w_i| \leq k$ , and  $|\beta_1 \dots \beta_n| > n \cdot k$ . Thus, if  $(S)^n \xrightarrow[G']{} (w_1, \dots, w_n)$ ,  $w_i$ ,  $1 \leq i \leq n$ , is in  $V_T^*$ ,  $0 < |w_1 \dots w_n| \leq k$ , then  $(w_1, \dots, w_n)$



will be in  $T_i$  for some  $i$ .

Also, observe that  $T_i \supset T_{i-1}$ , for all  $i \geq 1$ , and if  $T_i = T_{i-1}$ , then  $T_i = T_{i+1} = T_{i+2} = \dots$

The only thing that remains to be shown is that the algorithm terminates. We note that if  $T_i \neq T_{i-1}$ , then the number of forms in  $T_i$  is at least one greater than the number of forms in  $T_{i-1}$ . Let the number of symbols in  $V_N \cup V_T$  be  $y$ . Then the number of forms  $((\beta_1, \kappa_1), \dots, (\beta_n, \kappa_n))$  |  $|\beta_1 \dots \beta_n| \leq n \cdot k$ , is less than or equal to  $y + (y)^2 + \dots + (y)^{n \cdot k}$ , and hence less than  $(y + 1)^{n \cdot k + 1}$ .

Hence the result.

Q.E.D.

Next, we come to the emptiness problem.

#### Theorem 4.2.2:

There is an algorithm for determining if the language generated by a given nCG is empty.

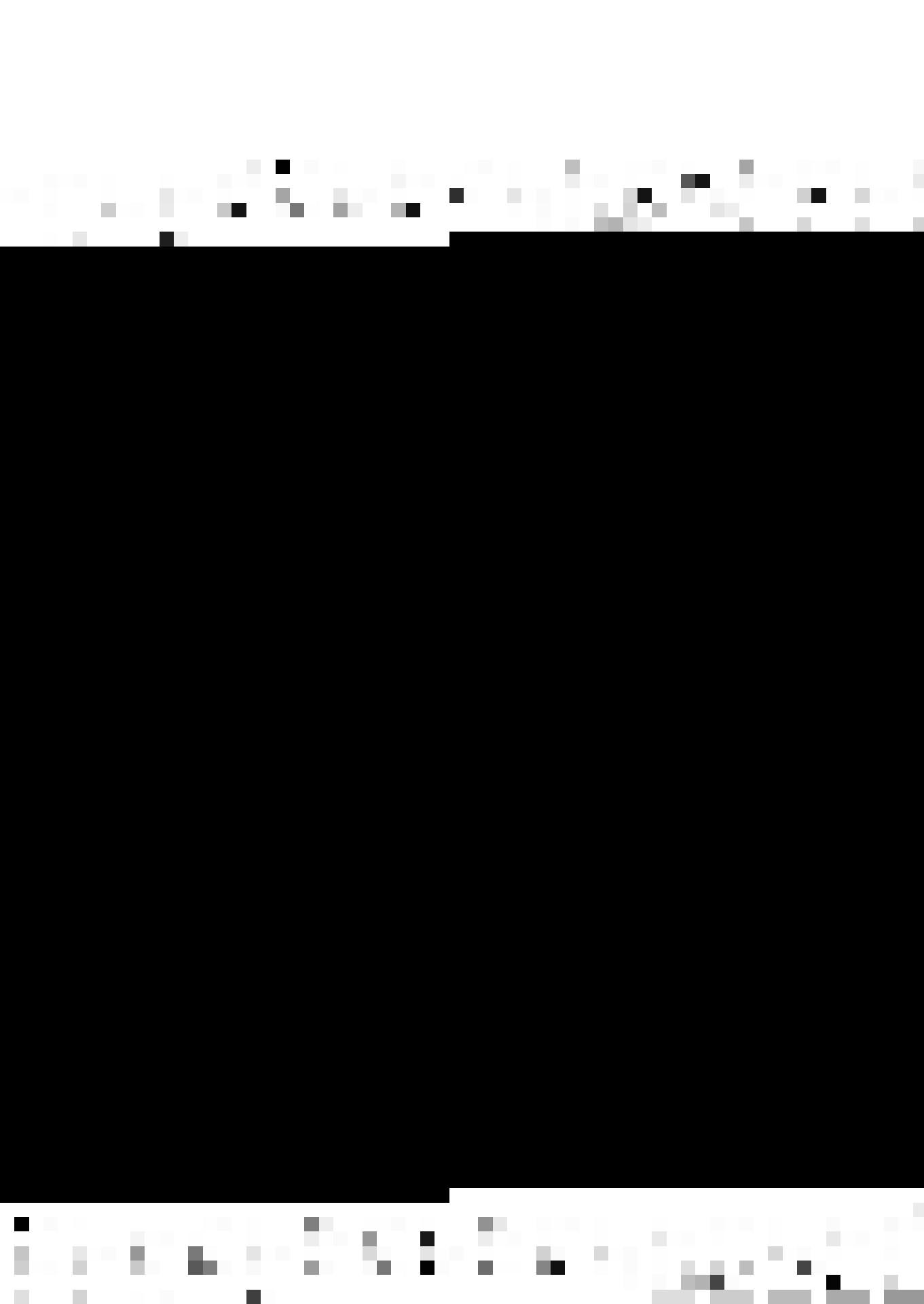
#### Proof:

Let  $G = (V_N, V_T, P, (S)^n)$ , be the given nCG. The following is the required algorithm. The proof of the correctness of the algorithm is simple and is omitted.

#### Algorithm:

We construct sets  $N_0, N_1, \dots$  recursively as follows:

1. Let  $N_0 = \emptyset$ . Set  $i = 1$ .



2. Let  $N_i = \left\{ A \mid A \rightarrow (\alpha_1, (\alpha_2, \overline{\alpha_2}), \dots, (\alpha_n, \overline{\alpha_n})), \right.$   
is in  $P$ , where the nonterminals occurring  
in  $\alpha_j$ ,  $1 \leq j \leq n$ , (if any) are in  
 $\left. N_{i-1} \right\} \cup N_{i-1}$ .

3. If  $N_i \neq N_{i-1}$ , then set  $i = i + 1$ , and go to  
step 2. Otherwise, let  $N_e = N_i$ .

4. If  $S$  is in  $N_e$ , then  $L(G)$  is not empty. Otherwise  
 $L(G)$  is empty.

Q.E.D.

The algorithm for testing whether  $\epsilon$  is in the language generated by a given nCG is similar to the algorithm given in the proof of the above theorem (see Appendix H).

Thus, we have:

#### Theorem 4.2.3:

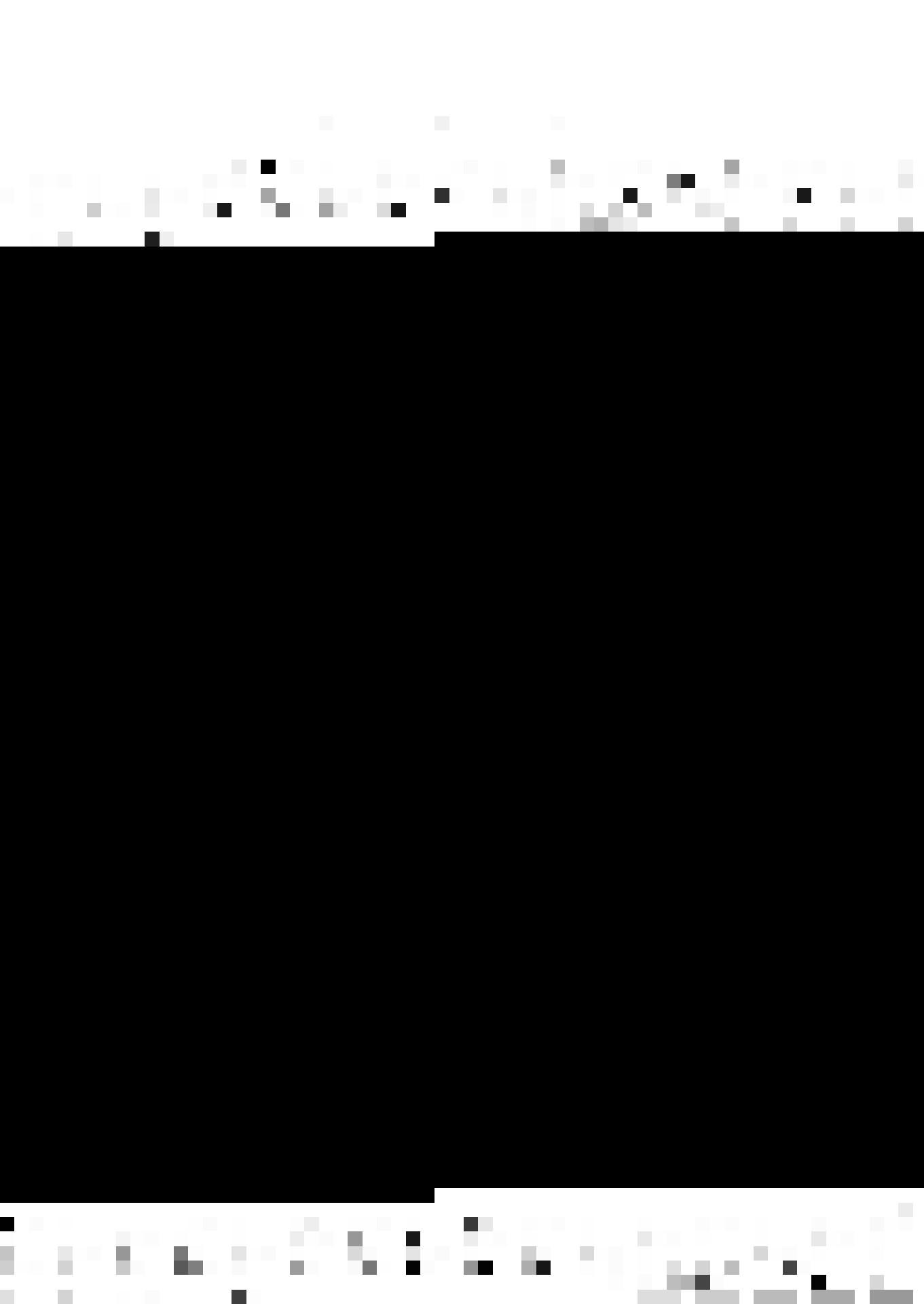
There is an algorithm to test if the language generated by a given nCG contains  $\epsilon$ .

\*\*\*

The following theorem establishes that the finiteness, infiniteness problem of nCG's is decidable.

#### Theorem 4.2.4:

There is an algorithm to determine if a given nCG  $G$  generates a finite or infinite number of sentences.



Proof:

Let  $p$  and  $q$  be the constants defined in the proof of Theorem 2.1.1. Thus (see Cor. 2.1.1.1) if  $z$  is in  $L(G)$  and  $|z| > n.p$ , then  $z$  can be written as:

$$u_1 v_1^i w_1 x_1 y_1 \dots u_n v_n^i w_n x_n y_n,$$

where for each  $i \geq 0$ ,  $u_1 v_1^i w_1 x_1^i y_1 \dots u_n v_n^i w_n x_n^i y_n$ , is in  $L(G)$ . Also,  $|v_1| + |x_1| + \dots + |v_n| + |x_n| > 0$ . Hence if there is a sentence of length greater than  $n.p$  in  $L(G)$ , then the language is infinite.

Suppose that  $L(G)$  is infinite. Then there are arbitrarily long sentences in  $L(G)$  and, in particular, a sentence of length greater than  $n.p + n.q$ . This sentence may be written as:  $u_1 v_1^i w_1 x_1 y_1 \dots u_n v_n^i w_n x_n y_n$ , where for each  $j$ ,  $1 \leq j \leq n$ ,  $|v_j w_j x_j| \leq q$ ,  $|v_1| + |x_1| + \dots + |v_n| + |x_n| > 0$  and  $u_1 v_1^i w_1 x_1^i y_1 \dots u_n v_n^i w_n x_n^i y_n$  is in  $L$  for all  $i \geq 0$ . In particular,  $u_1 w_1 y_1 \dots u_n w_n y_n$  is in  $L$ . Observe that  $|u_1 w_1 y_1 \dots u_n w_n y_n| < |u_1 v_1^i w_1 x_1 y_1 \dots u_n v_n^i w_n x_n y_n|$  and  $|u_1 w_1 y_1 \dots u_n w_n y_n| > n.p$ . If  $|u_1 w_1 y_1 \dots u_n w_n y_n| > n.p + n.q$ , we repeat the procedure until we eventually find a sentence in  $L(G)$  of length  $l$ ,  $n.p < l \leq n.p + n.q$ .

Thus  $L(G)$  is infinite if and only if it contains a sentence of length  $l$ ,  $n.p < l \leq n.p + n.q$ .

Since we can test whether a given sentence is in  $L(G)$  (see Theorem 4.2.1), we have merely to test all sentences of length greater than  $n.p$  and less than or equal to  $n.p + n.q$ .



for membership in  $L(G)$ . If there is such a sentence, then  $L$  is clearly infinite; if not, then there are no sentences of length greater than  $n.p$  in  $L(G)$ , so  $L(G)$  is finite.

The required algorithm is obvious from above.

Hence the result.

Q.E.D.

Finally, let us take up some of the undecidable problems of nCL and nSCL.

In view of Cor. 2.2.1.5 and the fact that every context-free language is in  $\mathcal{L}_1^L$  and hence in  $\mathcal{C}_1^L$ , we have the following undecidable results which follow from the corresponding results for context-free grammars (33).

#### Theorem 4.2.5:

The following decision problems are undecidable. Let  $G$ ,  $G_1$ ,  $G_2$  be arbitrary nCG's (or nSCG's):

1.  $L(G_1) \subset L(G_2)$ .
2.  $L(G_1) = L(G_2)$ .
3.  $L(G)$  is regular.
  
  
  
4.  $G$  is unambiguous.



5.  $L(G)$  is unambiguous.
6.  $\overline{L(G)}$  is empty.
7.  $\overline{L(G)}$  is infinite.
8.  $\overline{L(G)}$  is regular.
9.  $\overline{L(G)}$  is context-free.
10.  $L(G_1) \cap L(G_2) = \emptyset$ .
11.  $L(G_1) \cap L(G_2)$  is infinite.
12.  $L(G_1) \cap L(G_2)$  is regular.
13.  $L(G_1) \cap L(G_2)$  is context-free.

#### 4.3 CONCLUSIONS:

In this chapter, we have investigated some of the closure properties and decision problems of coupled languages. Table 4.3.1 and Table 4.3.2 summarize most of the results that we have obtained. We note that if some decision problem of  $C_n^L$  (respectively,  $\mathcal{S}_n^L$ ) is undecidable, then this problem of  $C^L$  (respectively,  $\mathcal{S}^L$ ) too is undecidable. Thus some of the results listed in Table 4.3.2 are valid for  $C^L$  and  $\mathcal{S}^L$ , as well. We have also proved that  $C^L$  (or  $\mathcal{S}^L$ ) is closed under concatenation.



Table 4.3.1

Closure under	$\mathcal{C}_n^L$	$\mathcal{S}_n^L$
Union	Yes	Yes
Substitution by context-free languages	Yes	Yes
Intersection with regular sets	-	Yes
Intersection	No	No
Complementation	No	No
Concatenation with context-free languages	Yes	Yes
Concatenation with languages in $\mathcal{C}_2^L$	No	-
Concatenation with languages in $\mathcal{S}_2^L$	-	No



Table 4.3.2

Problem	$C_n^L$	$\mathcal{L}_n^L$
Recursiveness	D	D
Emptiness	D	D
Finiteness, infiniteness	D	D
Is $L(G_1) \subseteq L(G_2)$ ?	U	U
Is $L(G)$ regular?	U	U
Is $G$ unambiguous?	U	U
Is $L(G)$ unambiguous?	U	U
Is $\overline{L(G)}$ empty?	U	U
Is $\overline{L(G)}$ infinite?	U	U
Is $\overline{L(G)}$ regular?	U	U
Is $\overline{L(G)}$ context-free?	U	U
Is $L(G_1) \cap L(G_2) = \emptyset$ ?	U	U
Is $L(G_1) \cap L(G_2)$ infinite?	U	U
Is $L(G_1) \cap L(G_2)$ regular?	U	U
Is $L(G_1) \cap L(G_2)$ context-free?	U	U

D - Decidable

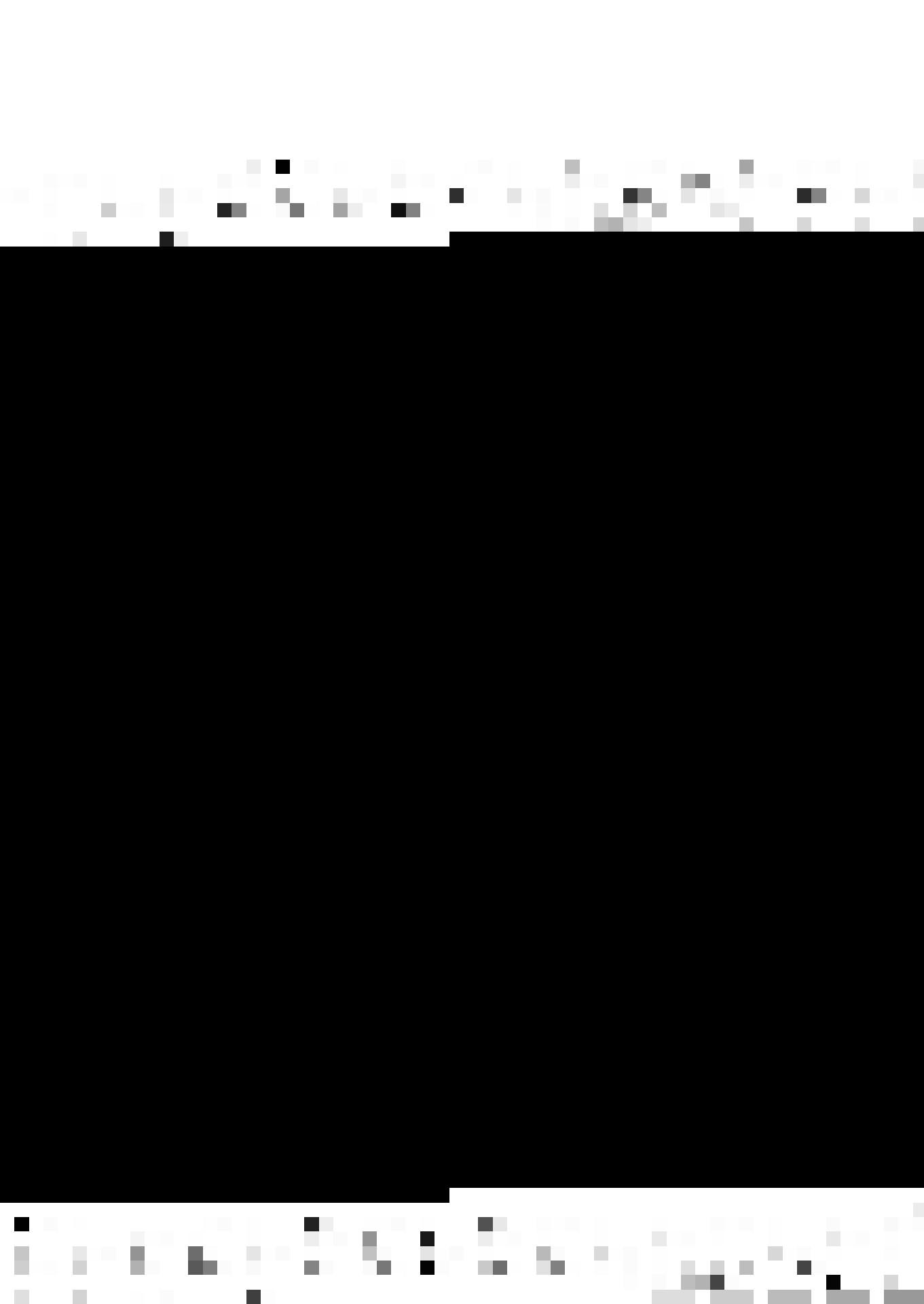
U - Undecidable



We have defined the concept of 'rank' for coupled languages which reveals some of their inherent properties. This concept is similar to that defined for equal matrix languages (36). We have proved that if  $L_1$  and  $L_2$  are coupled languages (or simple coupled languages) of rank  $m_1$  and  $m_2$ , respectively, then the rank of the language  $L = L_1 L_2$  is less than or equal to  $m_1 + m_2 - 1$ ; moreover,  $m_1 + m_2 - 1$  is a tight bound.

We have found the concept of n-derivation trees and the pumping theorem for coupled languages useful for proving some 'negative' results. The machine characterization of n-simple coupled languages has been useful for proving that n-simple coupled languages are closed under intersection with regular sets. We notice that such a result is significant because many other results follow from it.

We conjecture that n-coupled languages are closed under intersection with regular sets. A machine characterization of n-coupled languages will be quite useful in this regard. We also conjecture that  $C_n^L$  (or  $C^L$ ) is not closed under closure or  $\epsilon$ -free closure. Such a result will establish that  $C_n^L$  (or  $C^L$ ) is not an 'abstract family of languages' (15). Such a result is true for  $\mathcal{A}_n^L$  (or  $\mathcal{A}^L$ ) in view of the results in (19) and the fact that an n-simple coupled language is the same as simple matrix language of degree n (see Chapter 5). Because of the equivalence between  $\mathcal{A}_n^L$  and the class of simple matrix



languages of degree  $n$ , and that between  $\mathcal{R}_n^L$  and the class of equal matrix languages of order  $n$  (see Chapter 3), many additional results for  $\mathcal{S}_n^L$ ,  $\mathcal{S}^L$ ,  $\mathcal{R}_n^L$  and  $\mathcal{R}^L$  on closure properties and decision problems, become available.

The general implication of the results in this chapter is that though coupled languages form a richer class of languages than the class of context-free languages yet they have many properties in common. This is quite an encouraging fact and is in contrast to the case for classes of languages in the 'Chomsky hierarchy'.



## CHAPTER 5

### RELATIONSHIP OF THE CLASS OF COUPLED LANGUAGES WITH SOME KNOWN CLASSES OF LANGUAGES

In this chapter, we investigate the relationship of the class of coupled languages with some known classes of languages viz . the classes of context-sensitive languages, simple matrix languages (19) and equal matrix languages (34).

In section 5.1, we prove that the class of coupled languages is properly contained in the class of context-sensitive languages. In section 5.2, we show that the class of simple matrix languages of degree  $n$  is the same as the class of  $n$ -simple coupled languages. In section 5.3, we show that the class of equal matrix languages of order  $n$  is the same as the class of  $n$ -right linear coupled languages. We close the chapter with some conclusions and remarks in section 5.4.

#### 5.1 CONTEXT-SENSITIVE LANGUAGES AND COUPLED LANGUAGES:

In this section, we shall show that the class of coupled languages is properly contained in the class of context-sensitive languages (Theorem 5.1.1). In order to prove this result, we shall prove that any nCL is accepted by some deterministic linear bounded automaton (Lemma 5.1.1) and the Parikh mapping (see Appendix I) of an nCL is semilinear (Lemma 5.1.2).



Lemma 5.1.1:

Any nCL L is accepted by some deterministic linear bounded automaton.

Proof:

We will only discuss an outline of the proof<sup>1</sup>.

Let  $G = (V_N, V_T, P, (S)^n)$ , be an nCG in normal form such that  $L = L(G)$ . We will indicate how to construct from G a DLBA<sup>2</sup> which does 'analysis by synthesis' of the given language L. Before we discuss some details of the construction, we will consider a few points:

1. Let w be a sentence in L and  $|w| = m (> 0)$ . Let  $T = \langle T_1, \dots, T_n \rangle$ , be an n-derivation tree of w in G. Because G is in normal form, it is easy to see that the number of pendant nodes in each  $T_i$ ,  $1 \leq i \leq n$ , is less than or equal to m. From this and the fact that each nonpendant node in each  $T_i$ ,  $1 \leq i \leq n$ , has either got a single direct descendant which is a pendant node or has got two or more direct descendants which are nonpendant nodes, it follows that the derivation of w in G consists of  $2m-1$  or less steps. Without loss of generality, we can think of this derivation as a leftmost derivation (see Lemma 1.4.2).

1. The proof is a generalization of the proof of the corresponding result for context-free languages (24).
2. 'DLBA' stands for deterministic linear bounded automaton.



2. Let the number of productions in  $P$  be  $p$ . Let us associate numbers with these productions, from 0 to  $p-1$ . Any leftmost derivation in  $G$  can be described by a sequence of numbers in which the  $i$ -th number is the number associated with the production used in the  $i$ -th step of the derivation, provided at each step, track is kept of the  $n$  occurrences of a nonterminal which are to be expanded at the step. Clearly, the productions in  $P$  contain enough information for doing this 'tracking' work; note that there is no ambiguity at the first step of the derivation as to which nonterminals are to be expanded at this step. Let  $w$  be in  $L$  and let  $|w| = m$ . In view of 1 above, there cannot be more than  $2m-1$  numbers in the sequence that we have referred to above for describing the leftmost derivation of  $w$ , in  $G$  i.e. a  $(2m-1)$ -position base  $p$  number can be used to describe this derivation; with the  $i$ -th digit position from the left containing the number used in the  $i$ -th step of the derivation.

Let us now come to the construction of the required DLBA. There will be five tracks on the tape of the DLBA. The first track will hold the string being tested. The second track will hold the working string for the leftmost derivation. The third track will contain markers indicating which nonterminals ( $n$  occurrences of a nonterminal) are to be expanded in the next step of the derivation. The fourth track will hold



the base-p number which represents the choices of productions to be used in the derivation. The fifth track will contain a marker indicating which element of track 4 (i.e. which production) is to be used in the next step of the derivation.

To start a test, the machine writes the start form in track 2, places markers on track 3 against S's on track 2, writes a string of  $2m-1$  zeros on track 4 and places the marker on track 5 against the first position of the number on track 4. A typical configuration after this initialization process is shown in Fig. 5.1.1 for  $n = 3$ , and  $m = 4$ .

In view of Lemma 4.2.1, the maximum length of track that will be used while testing a string of length  $m$  will be  $nm + 2 + n - 1$  (excluding the spaces taken by the end markers), where  $2 + n - 1$  is the number of spaces taken for the commas and the two parentheses.

To simulate a single step of the derivation, the machine uses the following algorithm:

Algorithm:

- (i) Examine the symbol under the marker on track 4 to determine which production is to be used in the next step of the derivation. Then examine the nonterminals under the markers on track 2. Can the indicated production be used? If no, go to (iv), otherwise, continue.



						Track Numbers
*						5
C	0	0	0	0	*	4
*		*				3
(	S	,	S	S	)	2
a	b	a	a	*		1

16

The diagram shows three arrows originating from the bottom row of the table and pointing to the right. The first arrow points to the number 4. The second arrow points to the number 7. The third arrow points to the number 16.

Figure 5.1.1



- (ii) Make the transformations on track 2 (i.e. expand the form using the indicated production), as well as on track 3. (Clearly, from the information about the correspondence of nonterminals in the production used, the new positions of the markers on track 3 can be determined). Does the string on track 2 contain any nonterminal ? If yes, go to (vi), otherwise continue.
- (iii) Compare the string on track 2, ignoring the commas and parentheses, with that on track 1 (this is assuming that input string does not contain any comma or parenthesis; otherwise this can easily be taken care of). Does it match? If yes, the string is accepted, and go to (viii), otherwise continue.
- (iv) Is the number on track 4 equal to  $p^{2m-1} - 1$ ? If yes, the string under test is rejected and go to (viii), otherwise continue.
- (v) Increment the number on track 4 by 1. Replace the string on track 2 by the string  $(S)^n$ . Place the markers on track 3 against S's on track 2. Place the marker on track 5 against the first position of the number on track 4. Go to (i).
- (vi) Is the marker on track 5, at the rightmost available position (i.e. immediately to the left of \\$)? If yes, go to (v), otherwise continue.



(vii) Move the marker on track 5 right by one position.

Go to (i).

(viii) Stop.

Q.E.D.

Equipped with the concept of  $n$ -derivation trees, the proof of the following lemma is on the same lines as that of the corresponding result for context-free languages (28) and is given in Appendix I.

Lemma 5.1.2:

The Parikh mapping of an nCL is semilinear.

\*\*\*

Let  $\mathcal{C}_s$  denote the class of context-sensitive languages. From Lemmas 5.1.1 and 5.1.2, and a theorem in (24), it follows that :

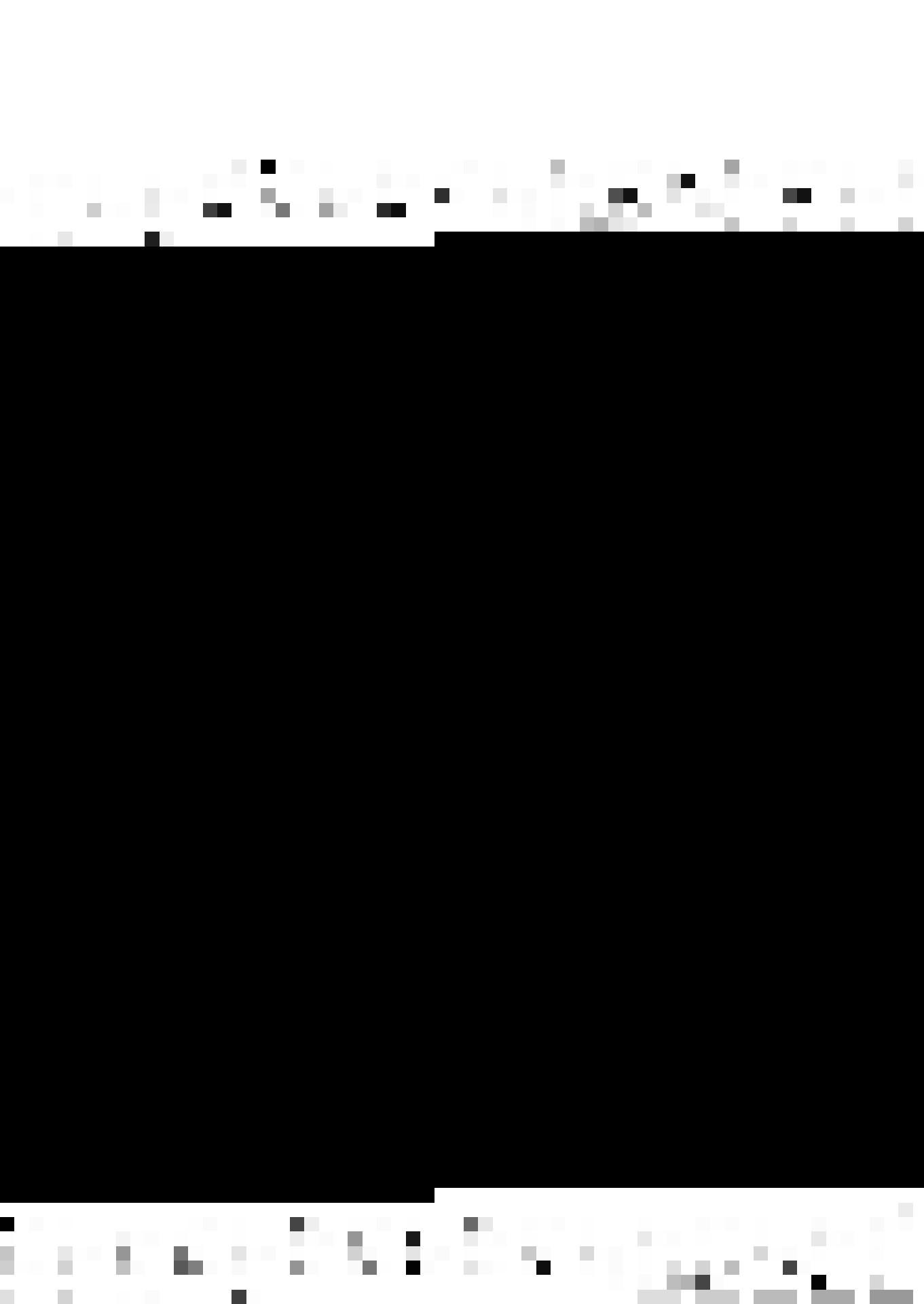
Theorem 5.1.1:

$$\mathcal{C}_s \supset \mathcal{C}^L$$

## 5.2 SIMPLE MATRIX LANGUAGES AND SIMPLE COUPLED LANGUAGES (39):

In this section, our aim is to show that the class of simple matrix languages of degree  $n$  (19) is the same as the class of  $n$ -simple coupled languages (Theorem 5.2.1).

We first give a few definitions concerning simple matrix languages which we have taken from Ibarra (19).

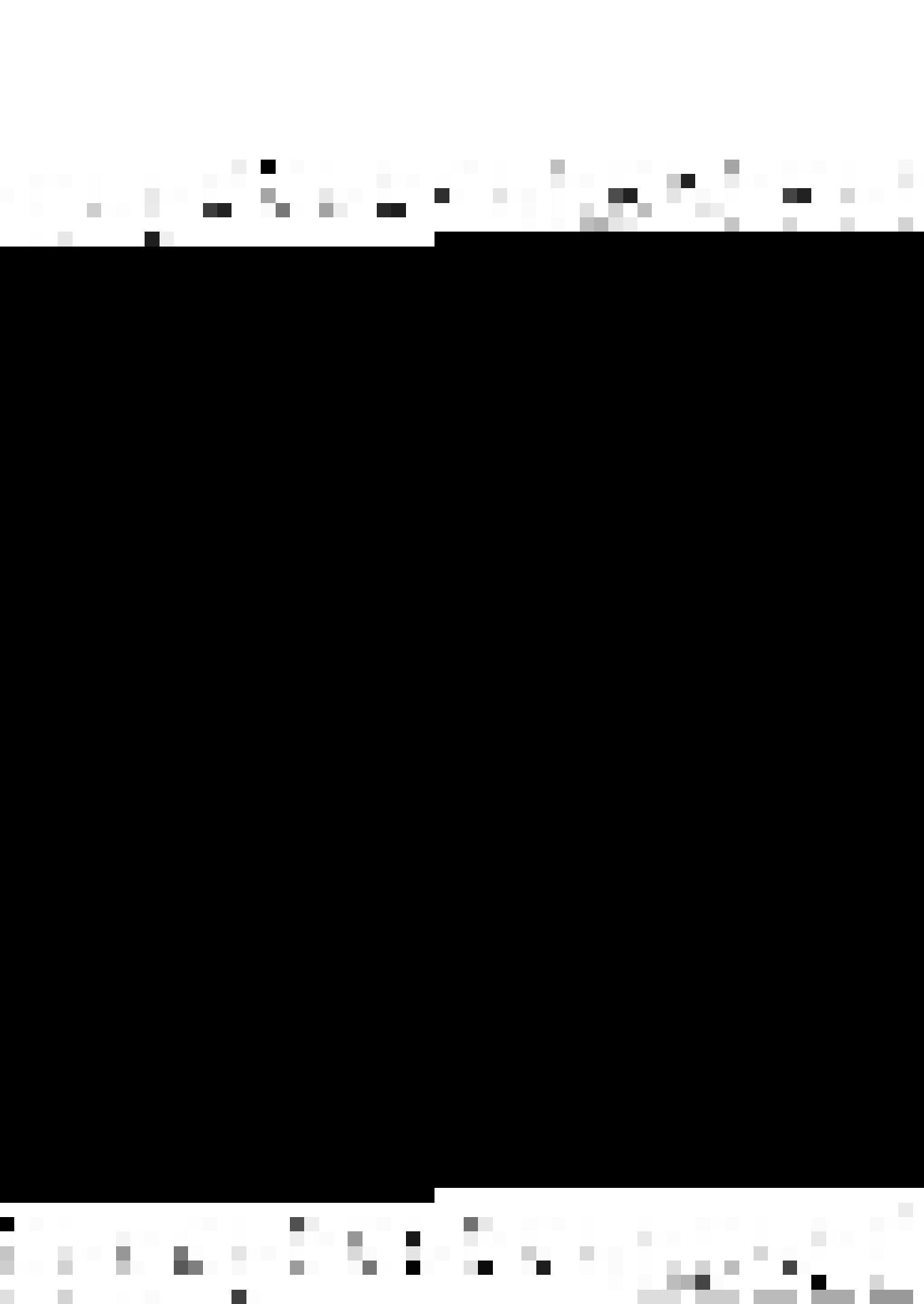


Let  $n \geq 1$ . A simple matrix grammar of degree n (or  $n$ -SMG) is an  $(n+3)$ -tuple  $G_n = \langle V_1, \dots, V_n, P, S, \Sigma \rangle$ , where

- (1)  $V_1, \dots, V_n$  are finite nonempty pairwise disjoint sets of nonterminal symbols;
- (2)  $\Sigma$  is a finite nonempty set of terminal symbols,  $\Sigma \cap V_j = \emptyset$ , for  $1 \leq j \leq n$ ;
- (3)  $S$  is not in  $V_1 \cup \dots \cup V_n \cup \Sigma$ , and is called the start symbol;
- (4)  $P$  is a finite set of matrix rewriting rules of the form:
  - (a)  $[S \rightarrow w]$ , where  $w$  is in  $\Sigma^*$ ,
  - (b)  $[S \rightarrow x_{11}^{A_{11}} x_{12}^{A_{12}} \dots x_{1k}^{A_{1k}} \dots x_{n1}^{A_{n1}} \dots x_{nk}^{A_{nk}} y]$ , where  $k \geq 1$ ,  $y$  is in  $\Sigma^*$ , and for  $1 \leq i \leq n$ ,  $1 \leq j \leq k$ ,  $A_{ij}$  is in  $V_i$  and  $x_{ij}$  is in  $\Sigma^*$ ,
  - (c)  $[A_1 \rightarrow w_1, \dots, A_n \rightarrow w_n]$ , where for  $1 \leq i \leq n$ ,  $A_i$  is in  $V_i$  and  $w_i$  is in  $\Sigma^*$ ,
  - (d)  $[A_1 \rightarrow x_{11}^{A_{11}} x_{12}^{A_{12}} \dots x_{1k}^{A_{1k}} y_1, \dots, A_n \rightarrow x_{n1}^{A_{n1}} \dots x_{nk}^{A_{nk}} y_n]$ , where  $k \geq 1$  and for  $1 \leq i \leq n$ ,  $1 \leq j \leq k$ ,  $y_i$ ,  $x_{ij}$  are in  $\Sigma^*$  and  $A_i, A_{ij}$  are in  $V_i$ .

\*\*\*

Let  $G_n = \langle V_1, \dots, V_n, P, S, \Sigma \rangle$  be an  $n$ -SMG. For  $\alpha, \beta$  in  $(V_1 \cup \dots \cup V_n \cup \Sigma \cup \{S\})^*$ , let  $\alpha \Rightarrow \beta$  if either (1) or (2) holds:



$w_i = x_{i1}^w i_1 x_{i2}^w i_2 \cdots x_{ik}^w i_k$ , and

$w_n = x_{n1}^w n_1 x_{n2}^w n_2 \cdots x_{nk}^w n_k y$ .

\*\*\*

The validity of the following result is easy to see.

Lemma 5.2.1:

With various symbols having the same meaning as in the above definition, if

$$x_{11}^A 1_1 x_{12}^A 1_2 \cdots x_{1k}^A 1_k \cdots x_{n1}^A n_1 \cdots x_{nk}^A n_k y \xrightarrow[G_n]{*} w,$$

then we can break up the derivation as follows:

$$x_{11}^A 1_1 x_{12}^A 1_2 \cdots x_{1k}^A 1_k \cdots x_{n1}^A n_1 \cdots x_{nk}^A n_k y$$

$$\xrightarrow[G_n]{*} x_{11}^w 1_1 x_{12}^A 1_2 \cdots x_{1k}^A 1_k \cdots x_{n1}^w n_1 \cdots x_{nk}^A n_k y$$

$$\xrightarrow[G_n]{*} x_{11}^w 1_1 x_{12}^w 1_2 \cdots x_{1k}^A 1_k \cdots x_{n1}^w n_1 x_{n2}^w n_2 \cdots x_{nk}^A n_k y$$

.

.

.

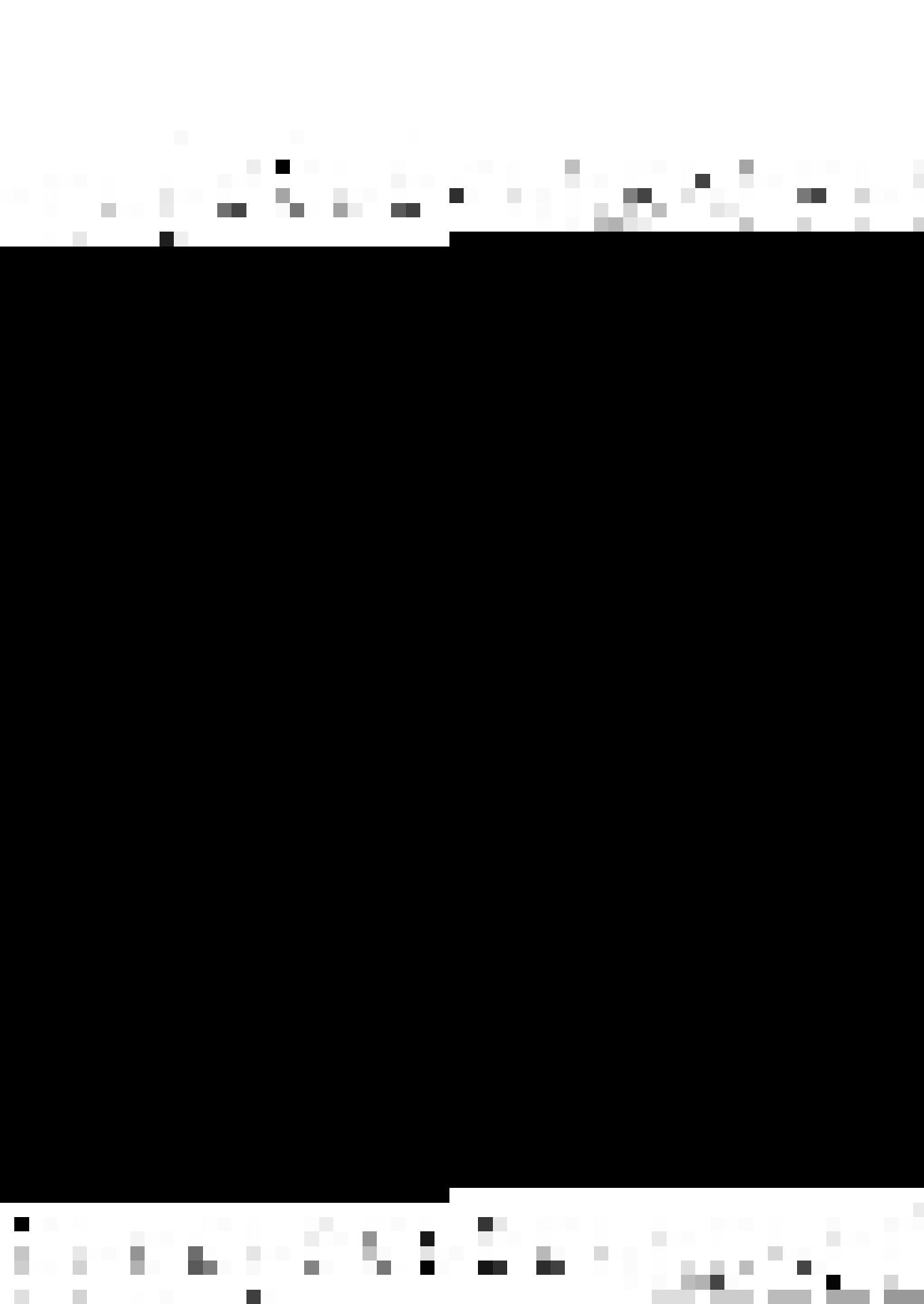
$$\xrightarrow[G_n]{*} x_{11}^w 1_1 x_{12}^w 1_2 \cdots x_{1k}^w 1_k \cdots x_{n1}^w n_1 x_{n2}^w n_2 \cdots x_{nk}^w n_k y$$

NOTATION:

$\mathcal{M}_n$  denotes the class of simple matrix languages of degree  $n$ .

\*\*\*

We now go about proving the main result of this section (Theorem 5.2.1) by proving Lemmas 5.2.2 and 5.2.3.



Lemma 5.2.2:

$$\mathcal{S}_n^L \subseteq \mathcal{M}_n$$

Proof:

Let  $G = (V_N, V_T, P, (S)^n)$ , be an nSCG in simple normal form, where  $V_N = \{A_{11}, \dots, A_{1k}\}$ . We construct an n-SMG  $G'_n$  such that  $L(G) = L(G'_n)$ , as follows:

$$G'_n = \langle V_1, \dots, V_n, P', S, V_T \rangle, \text{ where}$$

$$V_i = \{A_i, A_{i1}, A_{i2}, \dots, A_{ik}\}, 1 \leq i \leq n, \text{ and the}$$

rules of  $P'$  are constructed as follows:

(a) (i)  $[S \rightarrow A_1 A_2 \dots A_n]$ , is in  $P'$ .

(ii) For each nonterminating production:

$$S \rightarrow (A_{1p} A_{1q})^n, \text{ in } P, 1 \leq p, q \leq k, \text{ we have in } P'$$

the rule:

$$[A_1 \rightarrow A_{1p} A_{1q}, A_2 \rightarrow A_{2p} A_{2q}, \dots, A_n \rightarrow A_{np} A_{nq}]$$

(b) For each nonterminating production:

$$A_{1x} \rightarrow (A_{1p} A_{1q})^n, 1 \leq x, p, q \leq k, \text{ in } P, \text{ we have in } P'$$

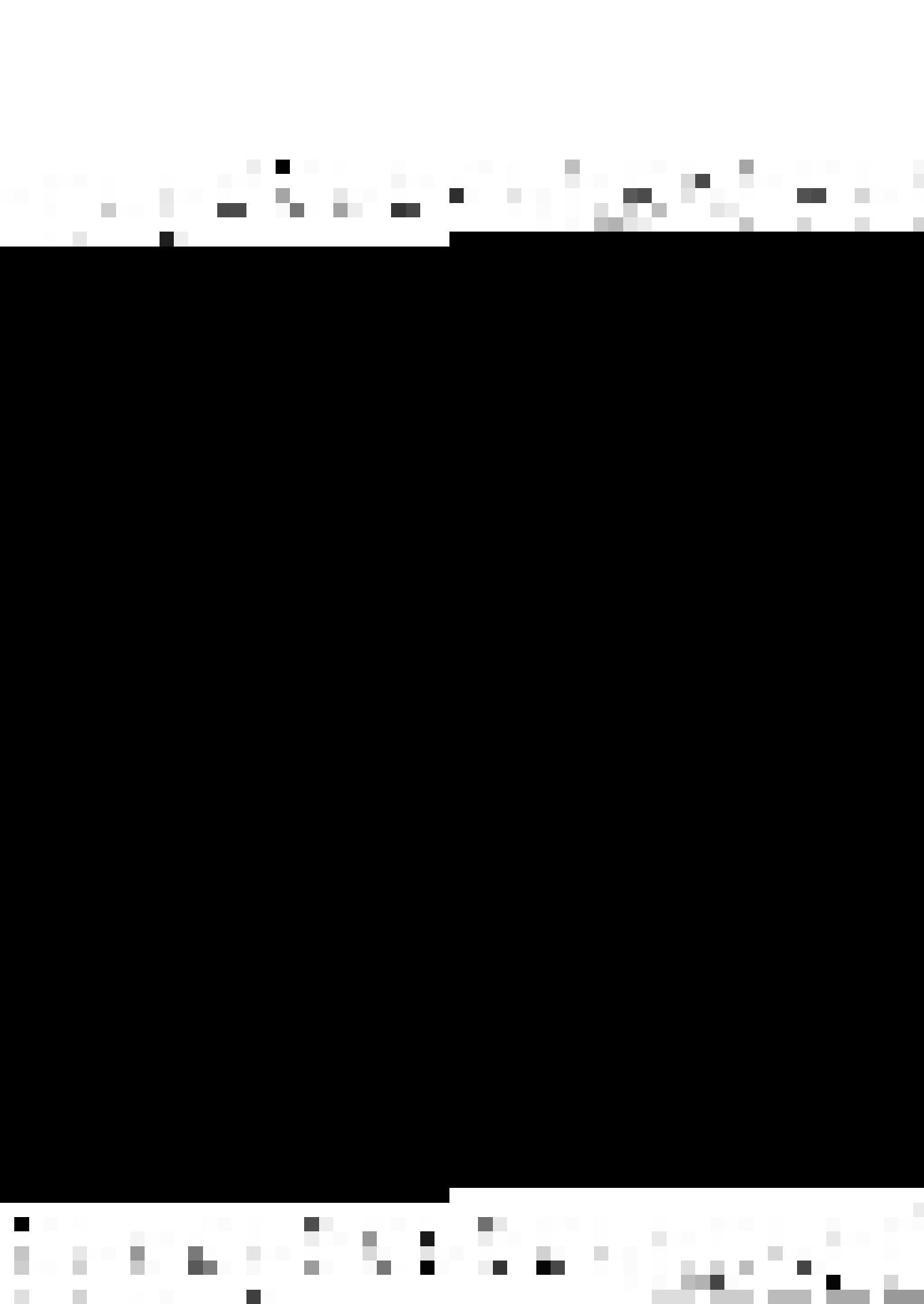
the rule:

$$[A_{1x} \rightarrow A_{1p} A_{1q}, A_{2x} \rightarrow A_{2p} A_{2q}, \dots, A_{nx} \rightarrow A_{np} A_{nq}]$$

(c) For each terminating production:

$$S \rightarrow (a_1, a_2, \dots, a_n), \text{ in } P, \text{ we have in } P' \text{ the rule:}$$

$$[S \rightarrow a_1 a_2 \dots a_n].$$



(d) For each terminating production:

$A_{1x} \rightarrow (a_1, a_2, \dots, a_n)$ ,  $1 \leq n \leq k$ , in  $P$ , we have in  $P'$

the rule:

$[A_{1x} \rightarrow a_1, A_{2x} \rightarrow a_2, \dots, A_{nx} \rightarrow a_n].$

\*\*\*

The idea behind the above construction is the following:

In  $n$ -simple coupled grammars, we have that all nonterminals which correspond with each other in a production are necessarily the same symbols. In  $n$ -SMG's this is not the case. We take care of this fact by creating  $n$  nonterminals  $A_{1i}, A_{2i}, \dots, A_{ni}$  in  $G'_n$ , for each nonterminal  $A_{1i}$  in  $G$ ,  $1 \leq i \leq k$ . For all purposes one can think of these nonterminals as 'corresponding' with each other. Again, in nSCG's there is nothing like a 'concatenating' production while there are 'concatenating' rules in  $n$ -SMG's. We let the 'corresponding with each other' nonterminals:  $A_1, \dots, A_n$ , serve the same function in  $G'_n$  as the start symbol  $S$  serves in  $G$ .

Now, we shall prove that  $L(G) = L(G'_n)$ . In order to do so, we shall first prove the following:

For  $i$ ,  $1 \leq i \leq n$ ,  $y_i, y'_i, w_i$  in  $V_T^*$ ,  $\alpha_i$  in  $(V_N \cup V_T)^*$ ,  $\beta_i$  in  $(V_i \cup V_T)^*$  and  $p$ ,  $1 \leq p \leq k$ ,  $(y_1^{A_{1p}} \alpha_1, \dots, y_n^{A_{1p}} \alpha_n)$

$\xrightarrow[G]{*} (y_1 w_1 \alpha_1, \dots, y_n w_n \alpha_n),$

iff

$$y_1^{A_{1p}} \beta_1 \dots y_n^{A_{np}} \beta_n \xrightarrow[G_n]{*} y_1^{w_1} \beta_1 \dots y_n^{w_n} \beta_n \quad (I)$$

We shall prove (I) by induction on the number of steps of derivation.

only if:

Suppose that the derivation in G is one step long.

This implies that there is a production:  $A_{1p} \rightarrow (w_1, \dots, w_n)$ , in P. From the above construction, it follows that the rule:  $[A_{1p} \rightarrow w_1, A_{2p} \rightarrow w_2, \dots, A_{np} \rightarrow w_n]$ , is in P'. Thus for one step derivations in G, the 'only if' part of (I) is true.

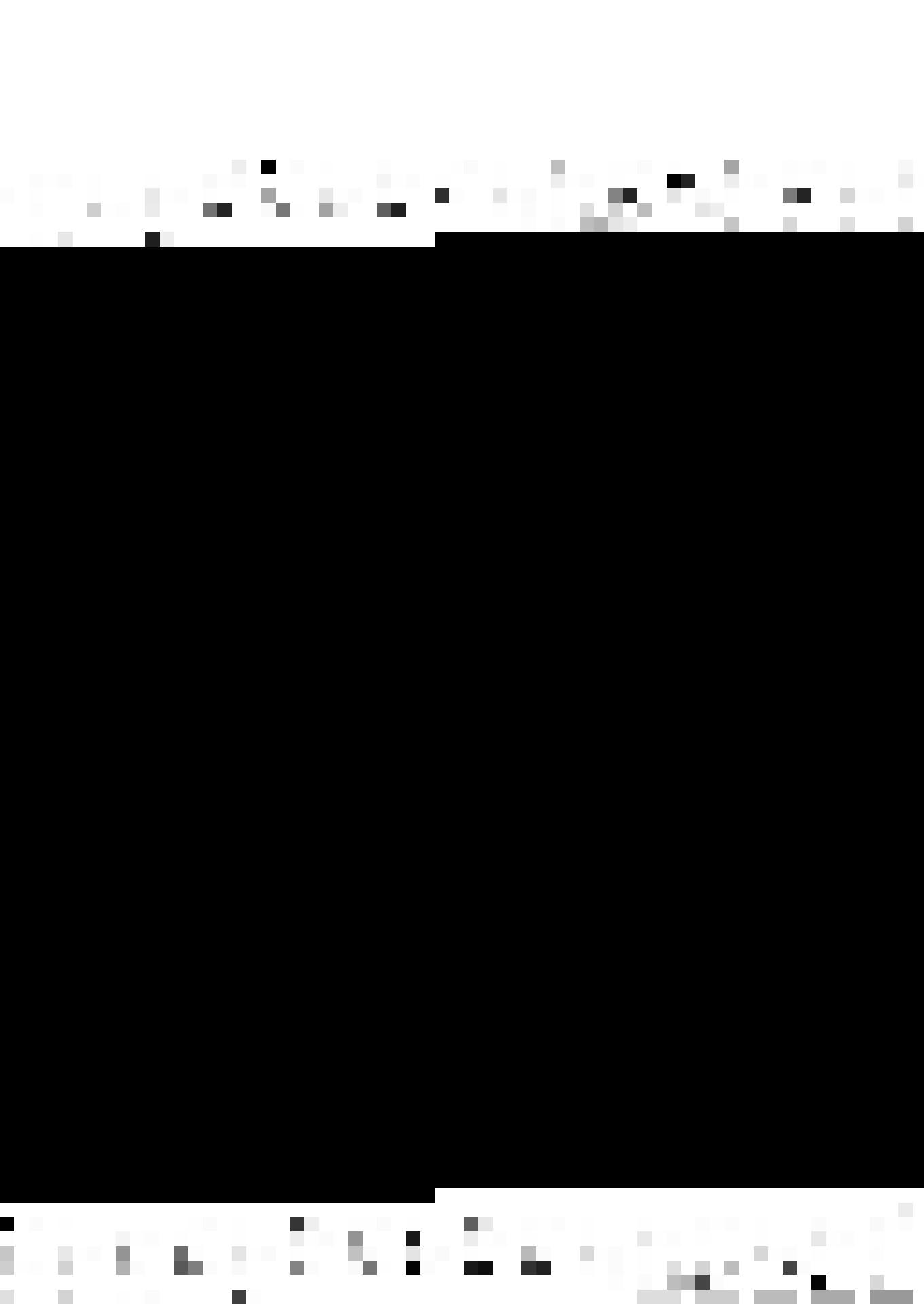
Assume as an induction hypothesis that the 'only if' part of (I) is true for derivations in G of length m or less, where  $m > 1$ .

Now, suppose that the derivation in G is of length  $m + 1$ . From Lemma 1.4.2, it follows that we can assume it to be a left-most derivation. Let the production used in the first step of the derivation be:  $A_{1p} \rightarrow (A_{1p}, A_{1q})^n$ ,  $1 \leq p', q' \leq k$ . Then,

$$(y_1^{A_{1p}} \alpha_1, \dots, y_n^{A_{1p}} \alpha_n)$$

$$\xrightarrow[G]{*} (y_1^{A_{1p}}, A_{1q}, \alpha_1, \dots, y_n^{A_{1p}}, A_{1q}, \alpha_n)$$

$$\xrightarrow[G]{*}^{(m \text{ steps})} (y_1^{w_1} \alpha_1, \dots, y_n^{w_n} \alpha_n).$$



From the construction of  $G'_n$ , it follows that there is a rule:

$$[A_{1p} \rightarrow A_{1p}, A_{1q}, \dots, A_{np} \rightarrow A_{np}, A_{nq}] \text{, in } P', \text{ and}$$

therefore,

$$y_1' A_{1p} \beta_1 \dots y_n' A_{np} \beta_n \xrightarrow[G'_n]{} y_1' A_{1p}, A_{1q}, \beta_1 \dots y_n' A_{np}, A_{nq}, \beta_n.$$

Now, use of the induction hypothesis completes the proof of this part.

The proof of the 'if' part of (I) is on the same lines as above and is omitted.

Now, let us come to the actual problem of proving that  $L(G) = L(G'_n)$ .

Let  $z$  be in  $L(G)$ . We shall prove that it is also in  $L(G'_n)$ . This is obvious from the construction of  $G'_n$ , if the derivation of  $z$  in  $G$  is one step long. If the derivation is more than one step long, let the first step of the derivation be:

$$(S)^n \xrightarrow[G]{} (A_{1p} A_{1q})^n.$$

From the construction of  $G'_n$ , we notice that the rules:

$[S \rightarrow A_1 A_2 \dots A_n]$ , and  $[A_1 \rightarrow A_{1p} A_{1q}, A_2 \rightarrow A_{2p} A_{2q}, \dots, A_n \rightarrow A_{np} A_{nq}]$ , will be in  $P'$ , so that  $S \xrightarrow[G'_n]^* A_{1p} A_{1q} A_{2p} A_{2q} \dots A_{np} A_{nq}$ . Now, using (I), it follows that  $z$  is in  $L(G'_n)$  also.



Again, let  $z$  be in  $L(G'_n)$ . We shall prove that it is in  $L(G)$  also. If the derivation of  $z$  in  $G'_n$  is one step long then we observe from the construction of  $G'_n$  that there will be a production:  $S \rightarrow (a_1, \dots, a_n)$ , such that  $z = a_1 \dots a_n$ . Thus  $z$  is also in  $L(G)$ . Now, let the derivation be more than one step long. In the first step of the derivation, the rule used has to be:  $[S \rightarrow A_1 A_2 \dots A_n]$ . Let the rule used in the second step be:

$$[A_1 \rightarrow A_{1p} A_{1q}, \dots, A_n \rightarrow A_{np} A_{nq}]$$

Corresponding to these rules in  $P'$ , there is the production:

$$S \rightarrow (A_{1p} A_{1q})^n, \text{ in } P. \text{ Thus,}$$

$$S \xrightarrow[G_n]{*} A_{1p} A_{1q} A_{2p} A_{2q} \dots A_{np} A_{nq}, \text{ and}$$

$(S)^n \xrightarrow[G]{*} (A_{1p} A_{1q})^n$ . That  $z$  is in  $L(G)$  also, follows now using (I).

Q.E.D.

### Lemma 5.2.3:

$$\mathcal{M}_n \subset \mathcal{B}_n^L$$

### Proof:

Let  $G_n = \langle V_1, \dots, V_n, P, S, \Sigma \rangle$ , be an  $n$ -SMG, where  $V_i = \{A_{i1}, A_{i2}, \dots, A_{ik_i}\}$ ,  $1 \leq i \leq n$ .



We construct an nSCG  $G'$  such that  $L(G_n) = L(G')$ , as follows:

$$G' = (V_N, \Sigma, P', (S)^n), \text{ where}$$

$$V_N = \left\{ ((A_1 \dots A_n)) \mid A_i \text{ is in } V_i, 1 \leq i \leq n \right\} \cup \{S\},$$

and  $P'$  is constructed in the following manner:

(a) For each rule:

$$[S \rightarrow w], \text{ in } P, \text{ we have in } P' \text{ the production: } S \rightarrow (w, \epsilon, \dots, \epsilon).$$

(b) For each rule:

$[S \rightarrow x_{11}A_{11} \dots x_{1k}A_{1k} \dots x_{n1}A_{n1} \dots x_{nk}A_{nk}y]$ , in  $P$ , we have in  $P'$  the production:

$$\begin{aligned} S \rightarrow & (x_{11}((A_{11} \dots A_{n1}))) \dots x_{1k}((A_{1k} \dots A_{nk})) \dots \\ & \dots, x_{n1}((A_{11} \dots A_{n1}))) \dots x_{nk}((A_{1k} \dots A_{nk}))y). \end{aligned}$$

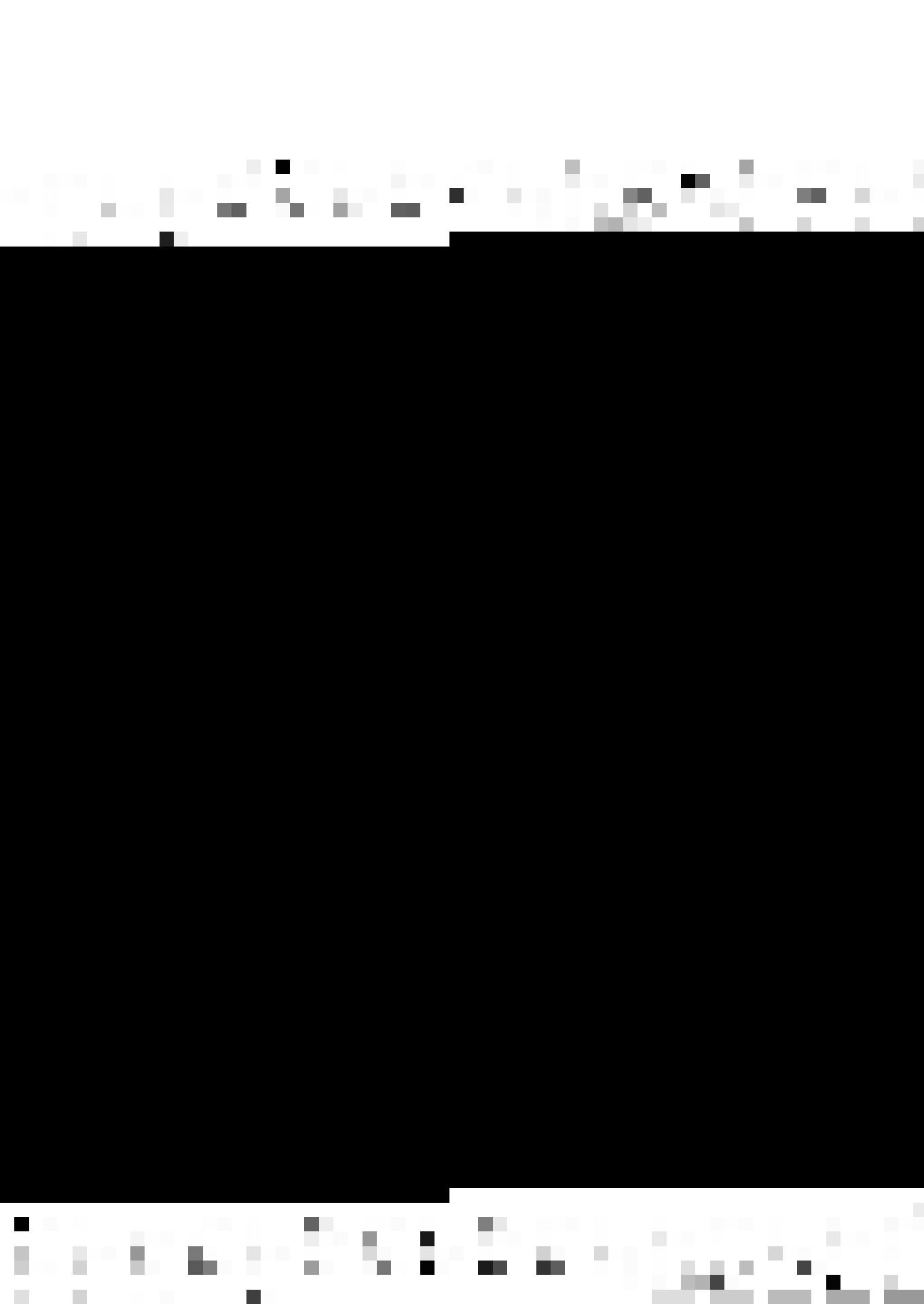
(c) For each rule:  $[A_{1i_1} \rightarrow w_1, \dots, A_{ni_n} \rightarrow w_n]$ , in  $P$ , we have in  $P'$  the production:

$$((A_{1i_1} \dots A_{ni_n})) \rightarrow (w_1, \dots, w_n).$$

(d) For each rule:

$$\begin{aligned} & [A_{1i_1} \rightarrow x_{11}A_{11} \dots x_{1k}A_{1k}y_1, \dots \\ & \dots, A_{ni_n} \rightarrow x_{n1}A_{n1} \dots x_{nk}A_{nk}y_n], \end{aligned}$$

in  $P$ , we have in  $P'$  the production:



$$\begin{aligned}
 ((A_{1i_1} \dots A_{ni_n})) \rightarrow & (x_{11}((A_{11} \dots A_{n1})) \dots x_{1k}((A_{1k} \dots A_{nk})), \dots \\
 & \dots, x_{n1}((A_{11} \dots A_{n1})) \dots x_{nk}((A_{1k} \dots A_{nk}))y).
 \end{aligned}$$

\*\*\*

The idea behind the construction of  $G'$  from  $G_n$  is to use a single nonterminal  $((A_1 \dots A_n))$  in  $G'$  such that the nonterminals  $A_1, \dots, A_n$  in  $G_n$  are used in a production in the same manner as 'corresponding' nonterminals are used in nSCG's, and to split the right hand sides of 'concatenating' rules in  $G_n$  to get the equivalent productions in  $G'$ .

Here also, we have:

For  $i, 1 \leq i \leq n$ ,  $y_i, y'_i, w_i$  in  $\Sigma^*$ ,  $\alpha_i$  in  $(V_N \cup \Sigma)^*$  and  $\beta_i$  in  $(V_i \cup \Sigma)^*$ ,

$$y_1^{A_{11}} \beta_1 y_2^{A_{21}} \beta_2 \dots y_n^{A_{n1}} \beta_n$$

$$\stackrel{*}{\underset{G_n}{\Rightarrow}} y_1^{w_1} \beta_1 y_2^{w_2} \beta_2 \dots y_n^{w_n} \beta_n$$

iff

$$\begin{aligned}
 (y'_1((A_{11} \dots A_{n1})) \alpha_1, \dots, y'_n((A_{11} \dots A_{n1})) \alpha_n) \\
 \stackrel{*}{\underset{G'}{\Rightarrow}} (y_1^{w_1} \alpha_1, \dots, y_n^{w_n} \alpha_n)
 \end{aligned} \tag{I}$$

(I) can be proved by induction on the no. of steps of derivation. The proof is on the same lines as that of (I) in Lemma 5.2.2, and is omitted.

Now, let us prove that  $L(G_n) = L(G')$ .



Let  $z$  be in  $L(G_n)$ . If the derivation of  $s$  in  $G_n$  is one step long, then there must be the rule:  $[S \rightarrow z]$ , in  $P$ . Then, from the construction of  $G'$ , it follows that there is a production:  $S \rightarrow (z, \epsilon, \dots, \epsilon)$ , in  $P'$ , so that  $(S)^n \xrightarrow[G']{*} (z, \epsilon, \dots, \epsilon)$ , which implies that  $z$  is in  $L(G')$  also. On the other hand, let the derivation of  $z$  in  $G_n$  be more than one step long. Let  $w_1 \dots w_n$  be the  $S$ -decomposition of  $z$  and let the rule used in the first step be:

$$[S \rightarrow x_{11}^{A_{11}} x_{12}^{A_{12}} \dots x_{1k}^{A_{1k}} \dots x_{n1}^{A_{n1}} \dots x_{nk}^{A_{nk}}].$$

From the construction of  $G'$ , it follows that the production:

$$S \rightarrow (x_{11}((A_{11} \dots A_{n1})) \dots x_{1k}((A_{1k} \dots A_{nk})), \dots$$

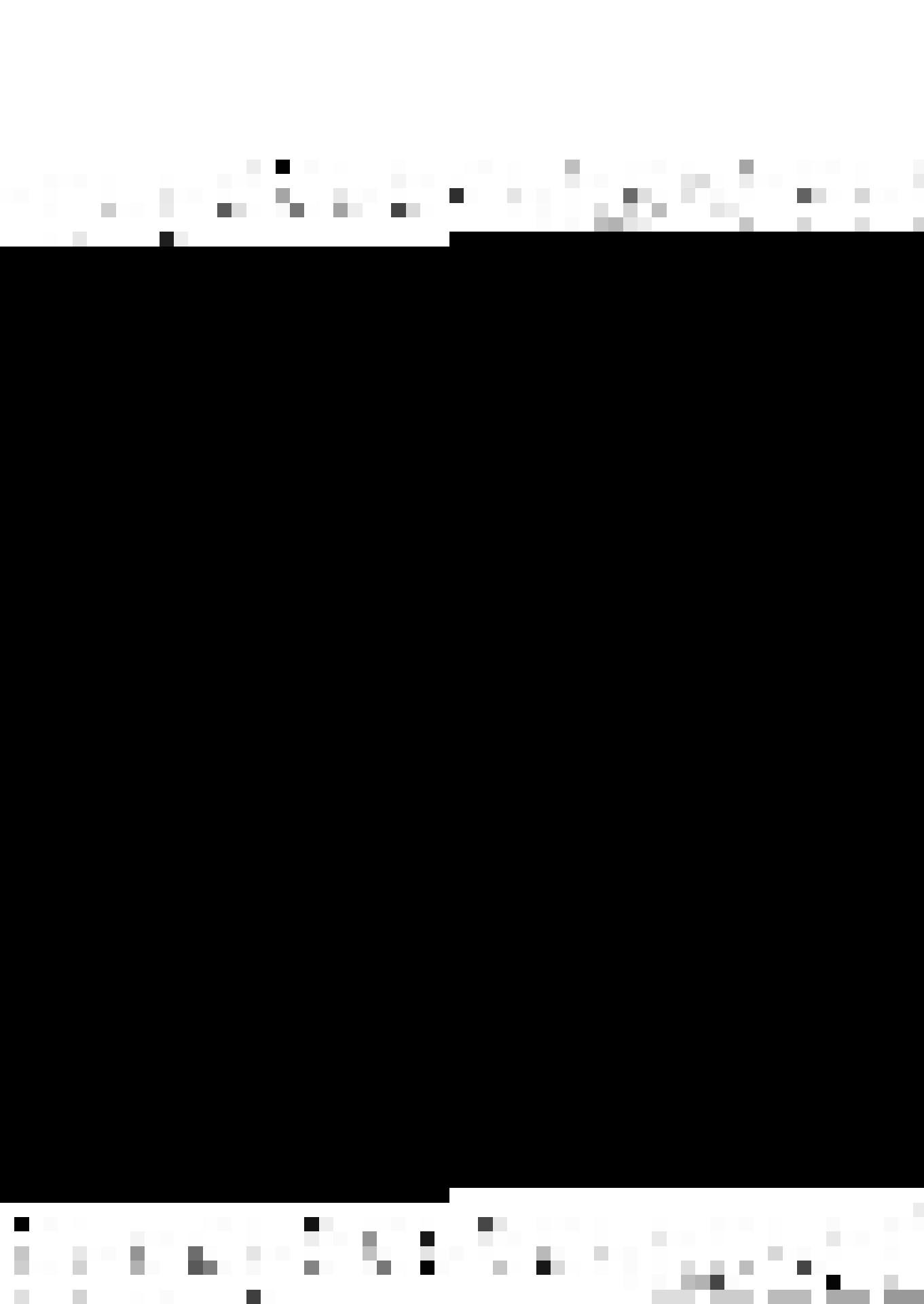
$$\dots, x_{n1}((A_{11} \dots A_{n1})) \dots x_{nk}((A_{1k} \dots A_{nk})))y), \text{ is in } P'.$$

Now, using Lemma 5.2.1 and (I) as above, it can be easily shown that  $(S)^n \xrightarrow[G']{*} (w_1, \dots, w_n)$ . The proof for it is omitted.

Again, let  $z$  be in  $L(G')$ . We shall show that it is also in  $L(G_n)$ . If the derivation of  $z$  in  $G'$  is one step long, then there must be the production:  $S \rightarrow (z, \epsilon, \dots, \epsilon)$ , in  $P'$ . Then, from the construction of  $G'$ , it follows that  $[S \rightarrow z]$ , is in  $P$ , so that  $z$  is in  $L(G_n)$  also. Now, let the derivation of  $z$  in  $G'$  be more than one step long. Let the first step of the derivation be:

$$(S)^n \xrightarrow[G']{*} (x_{11}((A_{11} \dots A_{n1})) \dots x_{1k}((A_{1k} \dots A_{nk})))y_1, \dots$$

$$\dots, x_{n1}((A_{11} \dots A_{n1})) \dots x_{nk}((A_{1k} \dots A_{nk})))y_n).$$



Considering the leftmost derivation of  $z$  in  $G'$ , the result follows using (I) as above.

Q.E.D.

From Lemmas 5.2.2, and 5.2.3, we have:

Theorem 5.2.1:

$$\mathcal{M}_n = \mathcal{S}_n^L$$

\*\*\*

From the above theorem and in view of Cor. 2.3.2.2, we have:

Theorem 5.2.2:

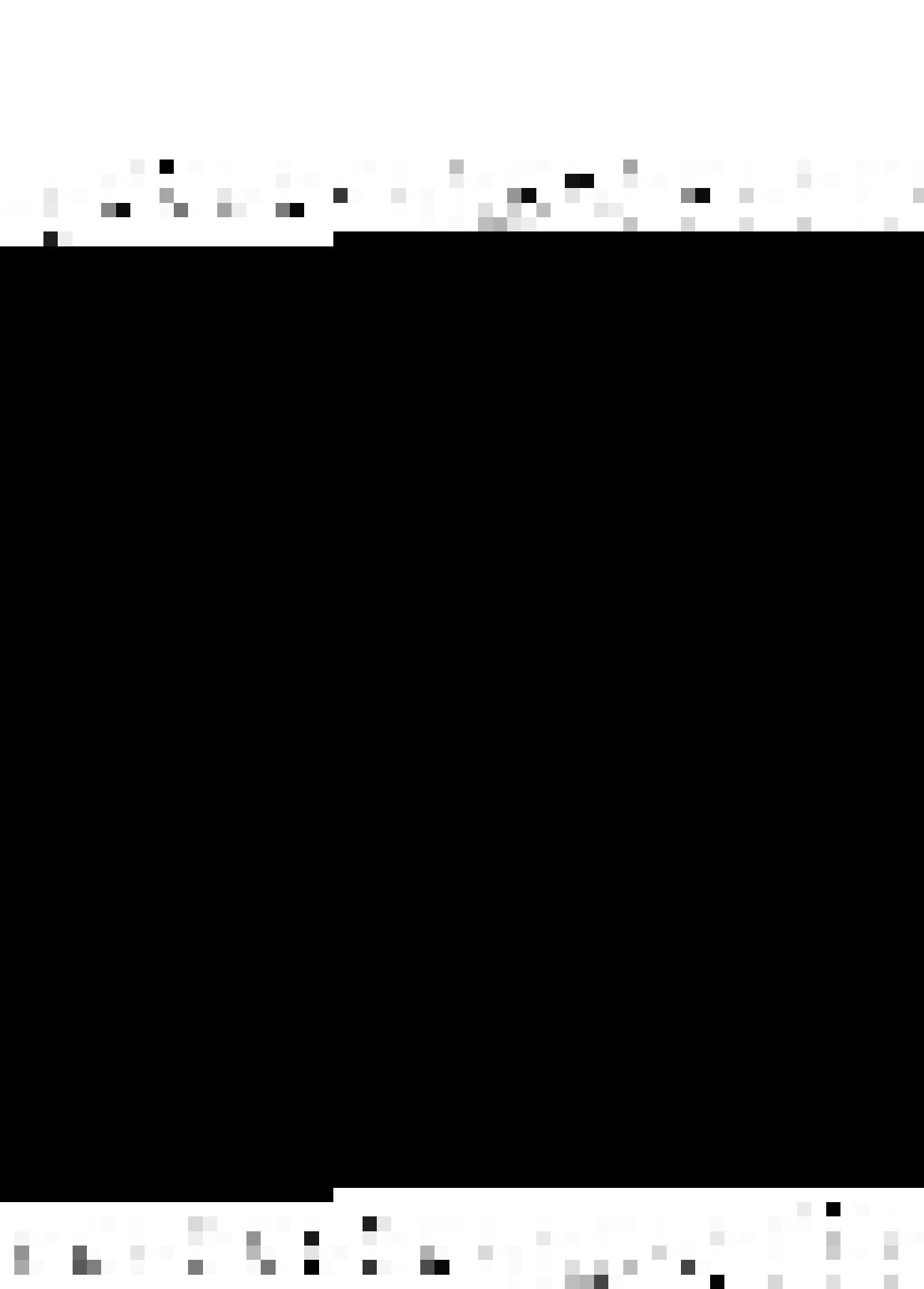
$$\mathcal{C}_n^L \supset \mathcal{M}_n$$

### 5.3 EQUAL MATRIX LANGUAGES AND RIGHT LINEAR COUPLED LANGUAGES:

The main result of this section is that the class of equal matrix languages of order  $n$  (34) is the same as the class of  $n$ -right linear coupled languages. It is to be noted that the class of equal matrix languages of order  $n$  is, by definition, the same class of languages as the class of right linear simple matrix languages of degree  $n$  (19).

Let us first give a few definitions concerning equal matrix languages which are taken from Siromoney (34).

$G = (V, I, P, S)$  is said to be an equal matrix grammar (EMG) of order  $n$  if



(1)  $V$  consists of the alphabet  $I$ , the initial symbol  $S$ , and the rest of the nonterminals  $V_N$  in the form of distinct  $n$ -tuples  $\langle A_1, \dots, A_n \rangle$ , and  $P$  consists of the following types of matrix rules:

(i) a set of initial rules of the form  $[S \rightarrow f_1 A_1 \dots f_n A_n]$ , where  $f_1, \dots, f_n$  are in  $I^*$ ,  $S$  the initial symbol, and  $\langle A_1, \dots, A_n \rangle$  in  $V_N$ .

(ii) a set of nonterminal rules of the form

$$\begin{bmatrix} A_1 & \rightarrow & f_1 B_1 \\ \vdots & & \vdots \\ A_n & \rightarrow & f_n B_n \end{bmatrix},$$

where  $f_1, \dots, f_n$  are in  $I^*$ ,  $\langle A_1, \dots, A_n \rangle$ ,  $\langle B_1, \dots, B_n \rangle$  are in  $V_N$ ,

(iii) a set of terminal rules of the form

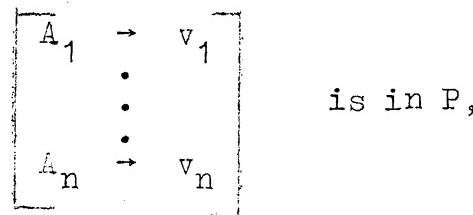
$$\begin{bmatrix} A_1 & \rightarrow & f_1 \\ \vdots & & \vdots \\ A_n & \rightarrow & f_n \end{bmatrix},$$

where  $f_1, \dots, f_n$  are in  $I^*$  and  $\langle A_1, \dots, A_n \rangle$  in  $V_N$ . \*\*\*

Let  $G = (V, I, P, S)$  be an EMG of order  $n$ . We write

$S \xrightarrow[G]{} f_1 A_1 \dots f_n A_n$  if  $[S \rightarrow f_1 A_1 \dots f_n A_n]$  is an initial rule in  $P$  and  
 $w_1 \xrightarrow[G]{} w_2$  if  $w_1 = x_1 A_1 \dots x_n A_n$ ,  $w_2 = x_1 v_1 \dots x_n v_m$  with  
 $x_1, \dots, x_n$  in  $I^*$ , and





and  $u \xrightarrow[G]{*} v$  is the reflexive, transitive closure of  $\xrightarrow[G]$ .

\*\*\*

$L \subset I^*$  is an equal matrix language (EML) if there is an EMG  $G = (V, I, P, S)$  such that  $L = L(G)$ , where  $L(G) = \{w \in I^* \mid S \xrightarrow[G]{*} w\}$ .  
 $L(G)$  is said to be the language generated by G.

NOTATION:

$\mathcal{E}_n$  denotes the class of equal matrix languages generated by EMG's of order n.

\*\*\*

The proof of the following theorem is similar to that of Theorem 5.2.1 (see Appendix J).

Theorem 5.3.1:

$$\mathcal{E}_n = \mathcal{R}_n^L$$

\*\*\*

From a result in (19), and Theorem 5.2.1, we have:

Theorem 5.3.2:

$$\mathcal{S}_n^L \supset \mathcal{E}_n$$

\*\*\*

From the above theorem and Theorem 5.3.1, we have:

Theorem 5.3.3:

$$\mathcal{S}_n^L \supset \mathcal{R}_n^L$$



#### 5.4 CONCLUSIONS:

In this chapter we have established the relationship of the class of coupled languages with some of the known classes of languages viz. context-free languages, context-sensitive languages, simple matrix languages and equal matrix languages. There is, however, scope for doing more work in this direction. A number of new classes of languages have been proposed in recent years, (2,17,21,30,31,32,40). It will be useful and interesting to find out how these classes are related to the class of coupled languages.

It follows from Lemma 1.2.1 and results in Chapter 2 that  $C_2^L$  (or  $\mathcal{S}_2^L$ ) properly contains the class of context-free languages. Thus  $C_n^L$  (or  $\mathcal{S}_n^L$ ),  $n \geq 1$ , defines an infinite hierarchy between the class of context-free languages and the class of context-sensitive languages. (While the hierarchy touches the class of context-free languages on one end, it does not touch the class of context-sensitive languages). One would like to know how these hierarchies are related to other similar hierarchies, for example that of Kasai (21).

In view of Theorem 5.2.1, all the work done for simple matrix languages (19) holds true for simple coupled languages. Similarly in view of Theorem 5.3.1, all the work done for equal matrix languages (34,35,36) holds true for right linear coupled languages.



## DISCUSSION

In this thesis, we have investigated a class of formal models for languages and translations which we call 'coupled grammars'. These grammars are generalizations of syntax directed translation schemes and can also be looked at as natural extensions of context-free grammars. As translation models these have been partly studied in (37). Our motivation for defining these grammars has been two fold:

(a) Need for a powerful class of models which, at the same time, are 'simple', and (b) Need for classes of models which represent naturally different types of structures. We have found that these grammars are, in fact, 'simple' in terms of the ease of writing these grammars for certain classes of languages and translations, and also in terms of their properties. Also, these grammars can represent naturally a class of structures which can be expressed through n-tuples of labeled rooted trees which are related to each other in a certain manner.

Coupled grammars can be used as language as well as translation modelling devices. One of the facts which makes these grammars powerful as translation models is that the domain and/or the range languages of these grammars can be non context-free languages. These domain and range languages are, in fact, coupled languages. This fact has prompted us



to concentrate on coupled languages in the thesis.

Where does the class of coupled languages stand in the Chomsky hierarchy? We have proved that it contains the class of context-free languages and is properly contained in the class of context-sensitive languages.

We have identified three 'complexity' parameters of coupled grammars: degree, order and  $\mu$ -simplicity. Out of these, the  $\mu$ -simplicity seems to be novel and interesting one. We have found that these parameters induce structures on the languages and translations generated by these grammars. We have obtained a number of results in this direction which are contained in Chapter 2.

In order to characterize these languages, we have defined a class of machines which we call  $n$ -turn generalized checking automata. We have proved that these machines characterize a subclass of coupled languages called simple coupled languages of degree  $n$  (Chapter 3). An important thing about the class of simple coupled languages of degree  $n$ , that we have found, is that it is the same as the class of simple matrix languages of degree  $n$  (Chapter 5). Thus we get a machine characterization of simple matrix languages of degree  $n$ . This is interesting in view of the fact that simple matrix grammars and simple coupled grammars look quite different, the former are, in fact, restricted matrix grammars.



The concept of n-derivation trees, which we have defined (Chapter 1), plays an important role in the thesis and we believe that it will play a significant role in the theory of coupled grammars.

The work on coupled grammars is far from complete. We have touched only a few issues. Some of the problems on which work can be done are:

- a) Characterization of coupled languages and coupled translations in various ways including through machines.
- b) The usefulness or otherwise of coupled grammars for modelling programming languages and their compilation.
- c) Application of coupled grammars in pattern recognition and analysis.
- d) Algorithms for efficient parsing of coupled grammars.
- e) Defining restricted coupled grammars which are efficient to handle; such work has been done for context-free grammars in recent years (11,13,22).



## REFERENCES

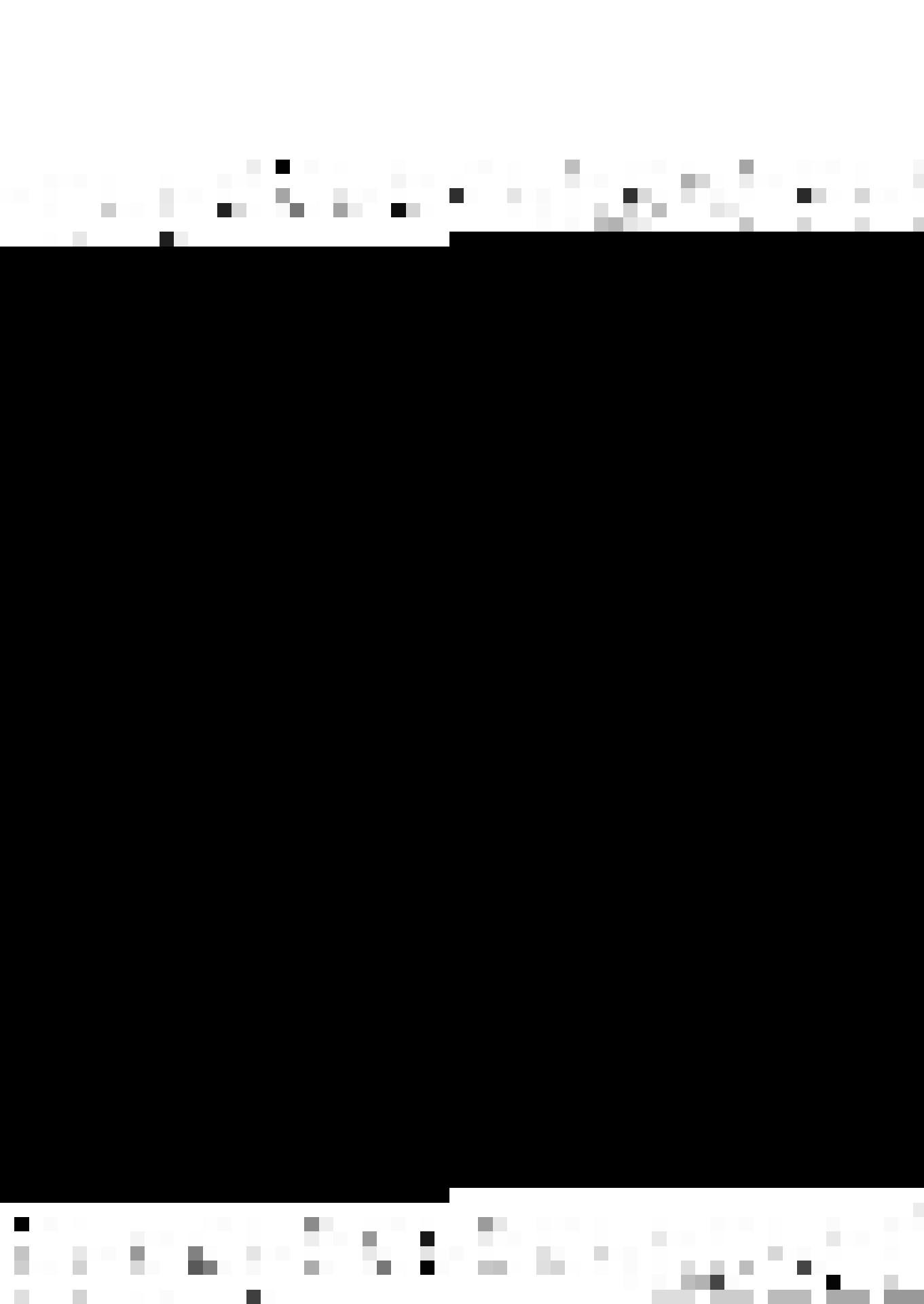
1. ABRAHAM, S. (1965)  
Some questions of phrase structure grammars.  
Computational Linguistics 4, 61-70.
2. AHO, A.V. (1968)  
Indexed grammars - an extension of context-free  
grammars.  
J. ACM 15:4, 647-671.
3. AHO, A.V., and J.D. ULLMAN (1969)  
Syntax directed translations and the pushdown  
assembler.  
J.CSS 3:1, 37-56.
4. AHO, A.V., and J.D. ULLMAN (1969)  
Properties of syntax directed translations.  
J.CSS 3:3, 319-334.
5. AHO, A.V., and J.D. ULLMAN (1971)  
Translations on a context-free grammar.  
IC 19:5, 439-475.
6. AHO, A.V., and J.D. ULLMAN (1971)  
Characterizations and extensions of pushdown  
translations.  
MST 5, 172-192.
7. AHO, A.V., and J.D. ULLMAN (1972)  
The Theory of Parsing, Translation and Compiling, Vol. 1.  
Prentice-Hall, Englewood Cliffs, N.J.
8. CHOMSKY, N. (1959)  
On certain formal properties of grammars.  
IC 2:2, 137-167.
9. CHOMSKY, N., and M.P. SCHUTZENBERGER (1963)  
The algebraic theory of context-free languages.  
In P. Braffort and D. Hirschberg (eds.),  
'Computer Programming and Formal Systems',  
North Holland Publ., Amsterdam, 118-161.
10. CHOMSKY, N. (1967)  
Foreword to M. Gross and A. Lentin,  
'Introduction to Formal Grammars',  
Springer-Verlag.



11. FELDMAN, J.A., and D. GRIES (1968)  
Translator writing systems.  
Comm. ACM 11:2, 77-113.
12. FLOYD, R.W. (1962)  
On the nonexistence of a phrase structure  
grammar for ALGOL 60.  
Comm. ACM 5:9, 483-484.
13. FLOYD, R.W. (1963)  
Syntactic analysis and operator precedence.  
J. ACM 10:3, 316-333.
14. GINSBURG, S. (1966)  
'The Mathematical Theory of Context-free Languages'.  
McGraw-Hill, New York.
15. GINSBURG, S., and S.A. GREIBACH (1969)  
Abstract families of languages.  
Memoir American Math. Soc. No. 87, 1-32.
16. GREIBACH, S.A. (1969)  
Checking automata and one-way stack languages.  
J. CSS 3, 196-217.
17. GREIBACH, S., and J. HOPCROFT (1969)  
Scattered context grammars.  
J. CSS 3, 233-247.
18. HOPCROFT, J.E., and J.D. ULLMAN (1969)  
'Formal Languages and their Relation to 'AUTOMATA'.  
Addison-Wesley, Mass.
19. IBARRA, O.H. (1970)  
Simple matrix languages.  
IC 17, 359-394.
20. IRONS, E.T. (1961)  
A syntax directed compiler for ALGOL 60.  
Comm. ACM 4:1, 51-55.
21. KASAI, T. (1970)  
An hierarchy between context-free and context-  
sensitive languages.  
J. CSS 4, 492-508.
22. KNUTH, D.E. (1965)  
On the translation of languages from left to right.  
IC 8:6, 607-639.
23. KNUTH, D.E. (1968)  
Semantics of context-free languages.  
MST 2:2, 127-146.



24. KURODA, S.Y. (1964)  
Classes of languages and linear bounded automata.  
IC 7, 207-223.
25. LEWIS, P.M., II, and R.E. STEARNS (1968)  
Syntax directed transduction.  
J. ACM 15:3, 464-488.
26. NAUR, P. (ed.) (1963)  
Revised report on the algorithmic language  
ALGOL 60.  
Comm. ACM 6:1, 1-17.
27. OGDEN, W. (1968)  
A helpful result for proving inherent ambiguity.  
MST 2:3, 191-194.
28. PARikh, R.J. (1966)  
On context-free languages.  
J. ACM 13:4, 570-581.
29. PETRONE, L. (1968)  
Syntax directed mapping of context-free languages.  
'IEEE Conference Record of Ninth Annual Symposium  
on Switching and Automata Theory', 176-186.
30. RAJLICH, V. (1972)  
Absolutely parallel grammars and two-way finite state  
transducers.  
J.CSS 6, 324-342.
31. ROSENKRANTZ, D.J. (1968)  
Programmed grammars and classes of formal languages.  
J. ACM 16:1, 107-131.
32. SALOMAA, A. (1970)  
Periodically time-variant context-free grammars.  
IC 17, 294-311.
33. SALOMAA, A. (1973)  
'Formal Languages'.  
Academic Press, New York.
34. SIROMONEY, R. (1969)  
On equal matrix languages.  
IC 14, 131-151.
35. SIROMONEY, R. (1971)  
Finite-turn checking automata.  
J.CSS 5:6, 549-559.



36. SIROMONEY, R. (1971)  
Equal matrix languages and Kleene closure.  
The Proceedings of the Indian Academy of  
Sciences LXXIV: 4 Sec. A, 172-188.
37. VAISHNAVI, V.K. (1971)  
'A Class of Formal Models for Translations'.  
M. Tech. Thesis, I.I.T. Kanpur (India).
38. VAISHNAVI, V.K. (1975)  
Formal models for languages and translations.  
'Proceedings of the 10-th Annual Convention of  
the Computer Society of India'.
39. VAISHNAVI, V.K., and S.K. BASU. (1975)  
Simple coupled languages, simple matrix languages  
and n-turn generalized checking automata.  
To be presented at the 'Symposium on Circuits,  
Systems and Computers, Centre of Advanced Study  
in Radiophysics and Electronics, Calcutta (India)'.
40. VAN DER WALT, A.P.J. (1972)  
Random context languages.  
'Proceedings of IFIP Congress '71', North-  
Holland, Amsterdam, 66-68.

J.ACM - Journal of Association of  
Computing Machinery

Comm.ACM - Communication of  
Association of Computing  
Machinery

IC - Information and Control.

J.CSS- Journal of Computer and  
System Sciences.

MST - Mathematical Systems  
Theory.



## APPENDIX . A

In this appendix, we shall construct an  $(n+1, \sigma, k)$  CG,  $k \geq 1$ ,  $G$  such that  $L(G) = \{a_1^i \dots a_{2n+1}^i \mid i \geq 1\}$ . We shall consider two cases: (i)  $k = 1$ , and (ii)  $k \geq 2$ .

(i)  $k = 1$ :

Here,  $G = (V_N, \{S\}, \{a_1, \dots, a_{2n+1}\}, P, (S)^{n+1})$ , where  $P$  comprises of the following productions:

$$1. \quad S \rightarrow (a_1 Sa_2, \dots, a_{2n-1} Sa_{2n}, a_{2n+1} S),$$

$$2. \quad S \rightarrow (a_1 a_2, \dots, a_{2n-1} a_{2n}, a_{2n+1}).$$

(ii)  $k = 2$ :

Let  $G' = (V'_N, \emptyset, P', (S_1)^{n+1})$  be the sample<sup>1</sup>  $(n+1, \sigma, k)$  CG.

$G = (V'_N \cup \{S\}, \{a_1, \dots, a_{2n+1}\}, P, (S)^{n+1})$ , such that the symbol  $S$  is not in  $V'_N$ , and where  $P = P' \cup \{S \rightarrow (a_1 Sa_2, \dots, a_{2n-1} Sa_{2n}, a_{2n+1} S), S \rightarrow (a_1 a_2, \dots, a_{2n-1} a_{2n}, a_{2n+1})\}$ .

It is easy to see that in both cases: (i) and (ii),  $L(G) = \{a_1^i \dots a_{2n+1}^i \mid i \geq 1\}$ , and  $G$  is an  $(n+1, \sigma, k)$  CG.

1. We define the sample  $(n, \sigma, k)$  CG in the proof of Theorem 1.1.1.



## APPENDIX B

This appendix is a supplement to the proof of Lemma 2.3.3 and should be read in conjunction with it. Here, we will prove the statements  $s_1$  and  $s_2$ .

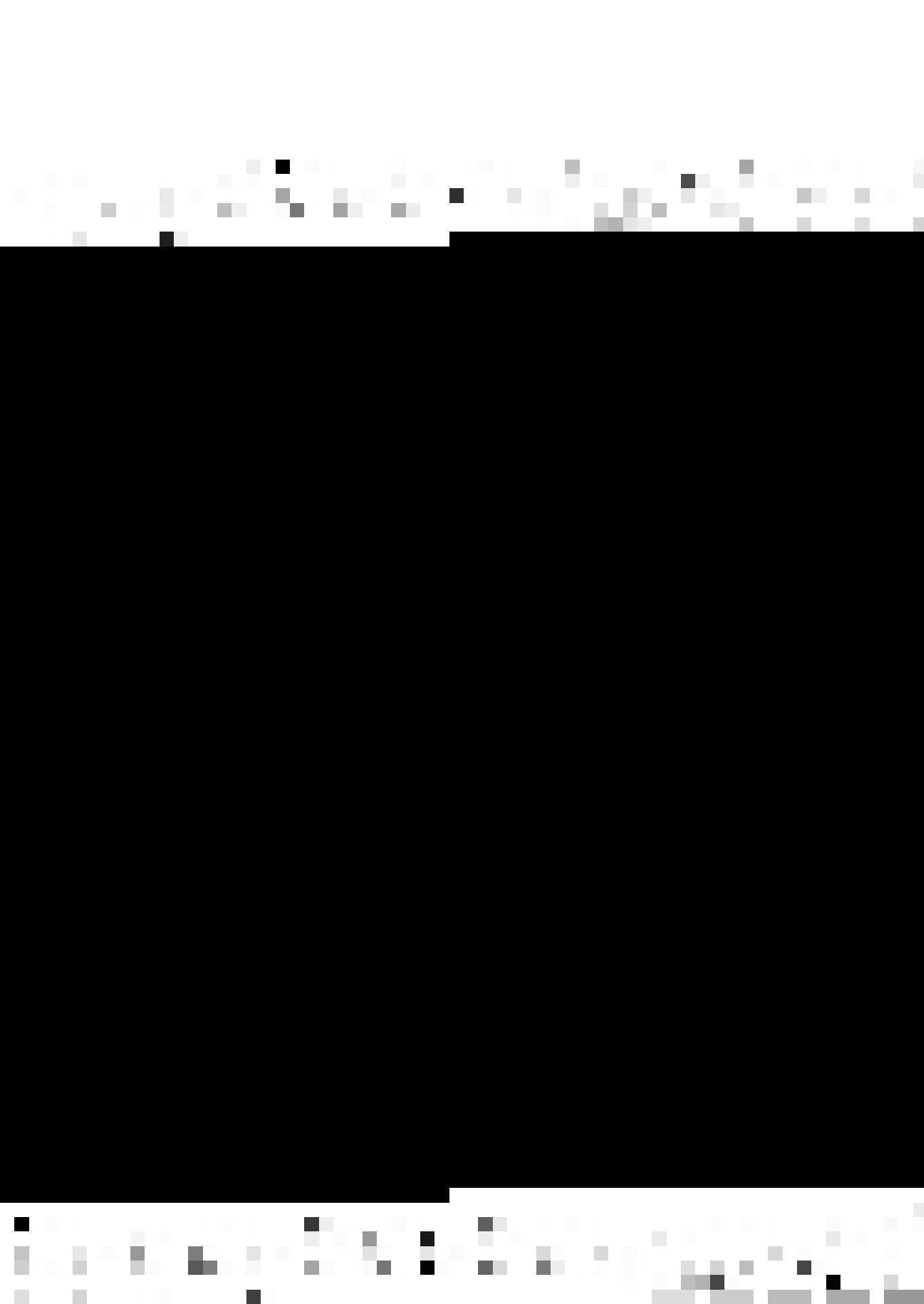
$s_1$ : Any element of  $X$  contains and contains only type  $(1,j)$  occurrences of  $a_{rm}$ , or contains and contains only type  $(1,j)$  occurrences of  $b_{rm}$ .

### Proof:

Consider some element  $x$  of  $X$ . We will assume that it contains a type  $(1,j)$  occurrence of  $a_{rm}$  (the proof in case it contains a type  $(1,j)$  occurrence of  $b_{rm}$  is similar and is omitted). We will prove that  $x$  does not contain an occurrence of a symbol different from  $a_{rm}$  and contains only type  $(1,j)$  occurrences of  $a_{rm}$ . Let us, for the time being, suppose to the contrary. This implies that either

1.  $x$  contains an occurrence of a symbol other than  $a_{rm}$ , or
2.  $x$  contains a type  $(s'', t'')$  occurrence of  $a_{rm}$ , where  $s'' \neq 1$  or  $t'' \neq j$ ,  $1 \leq s'' \leq n$ ,  $1 \leq t'' \leq n'$ .

In view of Lemma 2.3.1 (c and d), type  $(1,j)$  (or  $(s'', t'')$ ) occurrences of  $a_{rm}$  and  $b_{rm}$  are adjacent to each other, all such occurrences of  $a_{rm}$  preceding all such occurrences of  $b_{rm}$ , and so case 2 gets reduced to case 1.



Now, there can be two cases: (I) There is some symbol from among the symbols in  $Y = \{a_{11}, b_{11}, a_{12}, b_{12}, \dots, a_{(n-1)1}, b_{(n-1)1}, a_{(n-1)2}, b_{(n-1)2}\}$  such that  $x$  does not contain any occurrence of this symbol. (II)  $x$  contains an occurrence of each of the symbols in  $Y$ . We shall consider the two cases one by one.

(I): Since  $z(i)$  is in  $L$  for all  $i \geq 0$ ,  $z(3) = \alpha x x x \beta$ , for some  $\alpha$  and  $\beta$ , is also in  $L$ . This is a contradiction in view of Lemma 2.3.1 (c and d).

(II): Here again, there can be two cases: (i)  $x$  contains a type  $(1,j)$  occurrence of  $b_{rm}$  also; (ii)  $x$  does not contain any type  $(1,j)$  occurrence of  $b_{rm}$ .

(i): We observe that in  $z$ , all the type  $(1,j)$  occurrences of  $a_{rm}$  are adjacent to each other and are immediately followed by all the type  $(1,j)$  occurrences of  $b_{rm}$  (see Lemma 2.3.1(d)). Since  $x$  contains an occurrence of each of the symbols in  $Y$ , even in the worst case,  $x$  will contain  $a_{rm}^{n'} b_{rm}^q$  or  $a_{rm}^{n'} b_{rm}^q$ , as a substring, each of the two strings comprising of distinguished positions only (i.e. the occurrences of  $a_{rm}$  and  $b_{rm}$  in the two strings are type  $(1,j)$  occurrences of  $a_{rm}$  and  $b_{rm}$ , respectively). This implies that  $x$  contains a substring of length greater than  $q$ , which comprises of distinguished positions only. This is a contradiction in view of b'(ii).



(ii): This implies that  $x$  contains a type  $(s'', t'')$  occurrence of  $b_{rm}$ , where  $s'' \neq 1$  or  $t'' \neq j$ ,  $1 \leq s'' \leq n$ ,  $1 \leq t'' \leq n'$ , and the symbol  $g_{1j}$  does not occur in  $x$ . Since  $z(i)$  is in  $L$  for all  $i \geq 0$ ,  $z(2)$  is in  $L$ . Now, the contradiction follows in view of Lemma 2.3.1(b).

Hence the result.

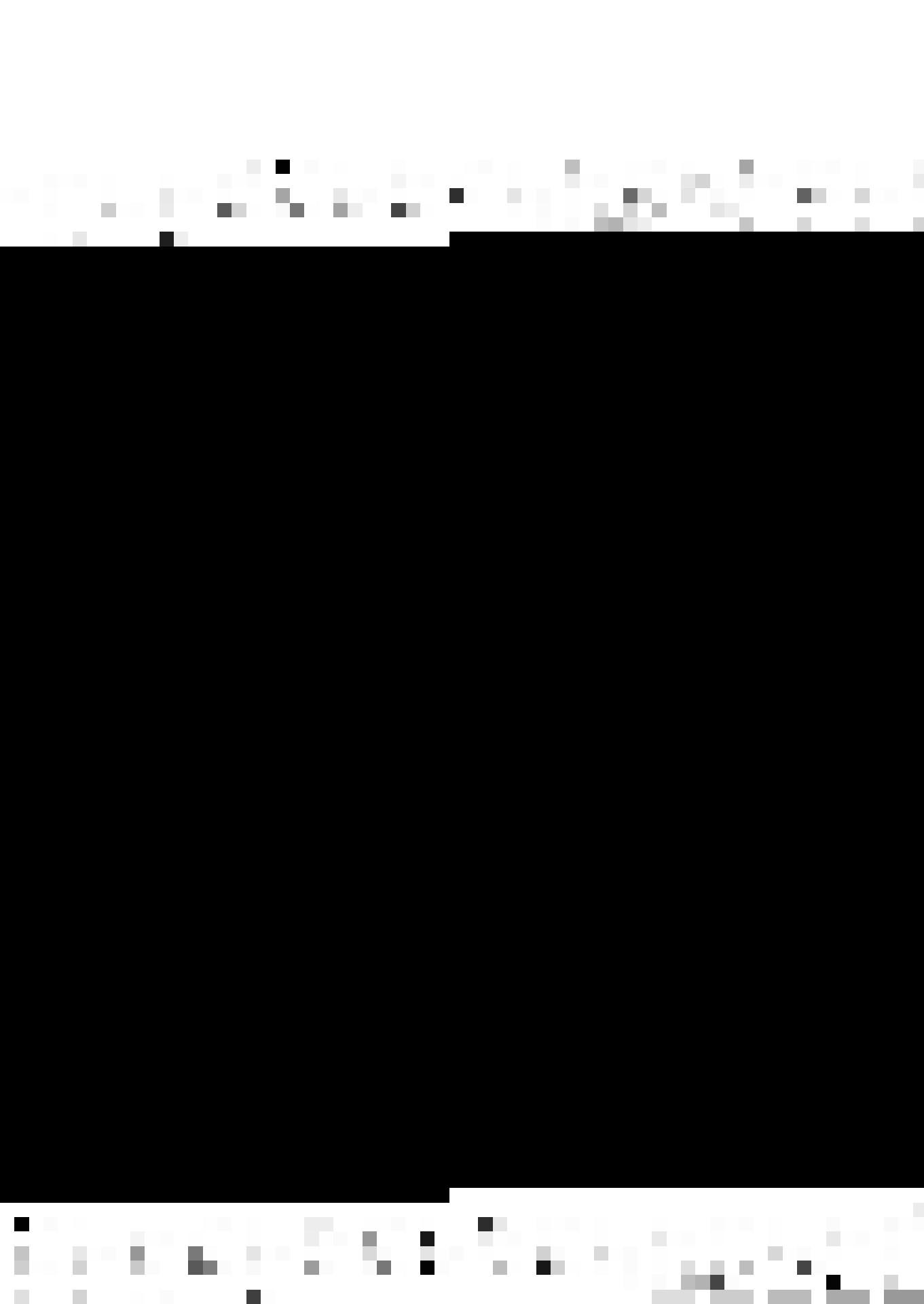
Q.E.D.

$s_2$ : A type  $(s', j)$ ,  $1 \leq s' \leq n$ , occurrence of  $a_{rm}$  (respectively,  $b_{rm}$ ) must be in  $V_{s'}$ , where  $V_{s'}$  is  $v_i$  or  $x_i$ ,  $1 \leq i \leq n'$ , such that  $V_{s'}$  contains and contains only type  $(s', j)$  occurrences of  $a_{rm}$  (respectively,  $b_{rm}$ ).

Proof:

Since  $X$  is not empty (in view of  $b'(i)$ ), we have: (i)  $V_1$  contains a type  $(1, j)$  occurrence of  $a_{rm}$ , or (ii)  $V_1$  contains a type  $(1, j)$  occurrence of  $b_{rm}$ . Let us assume that (i) is the case. In case (ii) is the case, the proof will be similar.

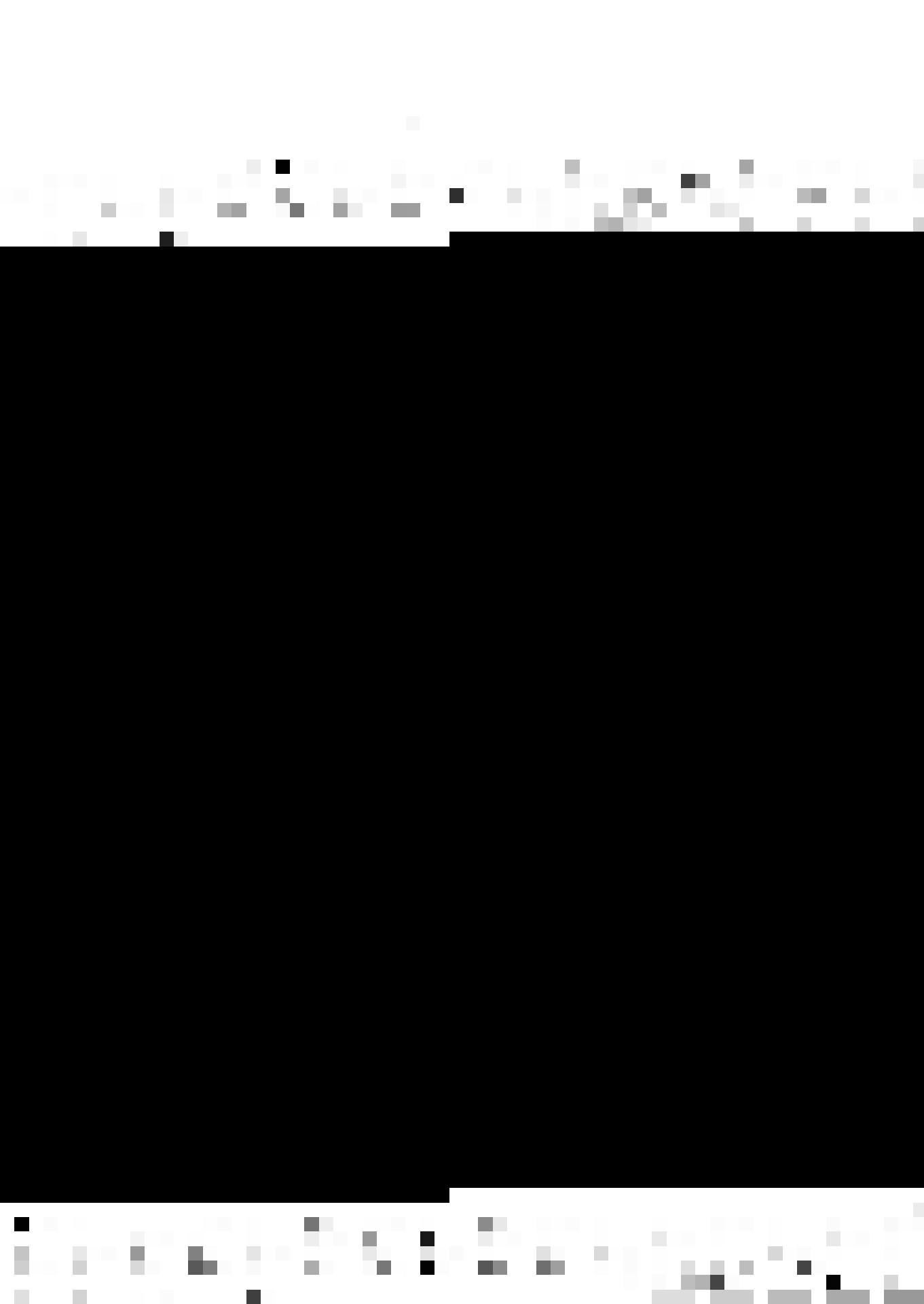
In view of  $s_1$ ,  $V_1$  contains and contains only type  $(1, j)$  occurrences of  $a_{rm}$ . Because  $z(i)$  is in  $L$ , for all  $i \geq 0$ ,  $z(2)$  must be in  $L$ . Now,  $z(2)$  has a segment of  $a_{rm}$  of length atleast  $n'.q+1$ . In view of Lemma 2.3.1(f), there are atleast  $n-1$  more segments of  $a_{rm}$  and atleast  $n$  segments of  $b_{rm}$ , each of length atleast  $n'.q+1$ . Note that in  $z$ , there are exactly  $n$  segments of  $a_{rm}$  and  $n$  segments of  $b_{rm}$ , each of length  $n'.q$  and the occurrences of symbols in these segments are of



type  $(s', j)$  for different values of  $s'$ ,  $1 \leq s' \leq n$ . Also note that the smallest value that  $q$  can have is 2. Now, it is easy to see that if the statement  $s_2$  is not true, then  $z(2)$  has not got atleast  $n$  distinct segments of  $a_{rm}$  and atleast  $n$  distinct segments of  $b_{rm}$ , each of length atleast  $n'.q+1$ . This is a contradiction.

Hence the result.

Q.E.D.



## APPENDIX C

This appendix supplements the proof of Lemma 2.3.4.

Here, we show that the language  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$  (defined in section 2.3) is in  $\mathcal{C}_{(n, \sigma, k)}^L$ , where  $n \geq 2$ ,  $k \geq 2$ ,  $n' \geq n$ , and  $\sigma = \xi_1, \dots, \xi_{n-1}$ . In view of Cor. 2.2.1.1, it suffices to show that  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$  is in  $\mathcal{C}_{(n, \sigma, 2)}^L$ .

We shall construct a  $(n, \sigma, 2)$ CG G such that G generates the language  $L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$ , with the help of the sample  $(n, \sigma, 2)$ CG (see the proof of Theorem 1.1.1).

$$G = (V_N, V_T, P, (S)^n), \text{ where}$$

$$V_N = \left\{ S, S', S_1, S_2, C, A, B_{11}, B_{12}, B_{21}, B_{22}, \dots, B_{(n-1)1}, B_{(n-1)2}, B_1, \dots, B_{n-1}, D_1, \dots, D_{n'}, F_1, \dots, F_{n'-2}, G_1, \dots, G_{n'-3}, H_1, \dots, H_{n-3} \right\},$$

$$V_T = \left\{ a_{ij} \mid 1 \leq i \leq n-1, 1 \leq j \leq 2 \right\} \cup \left\{ g_1, \dots, g_n \right\} \cup \left\{ g_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n' \right\} \cup \left\{ g'_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n' \right\}, \text{ and}$$

P comprises of the following productions:

$$1. \quad S \rightarrow (S' g_1, \dots, S' g_n),$$

$$2. \quad S' \rightarrow (S_1 S_2)^n,$$

$$3. \quad S' \rightarrow (S_2)^n,$$

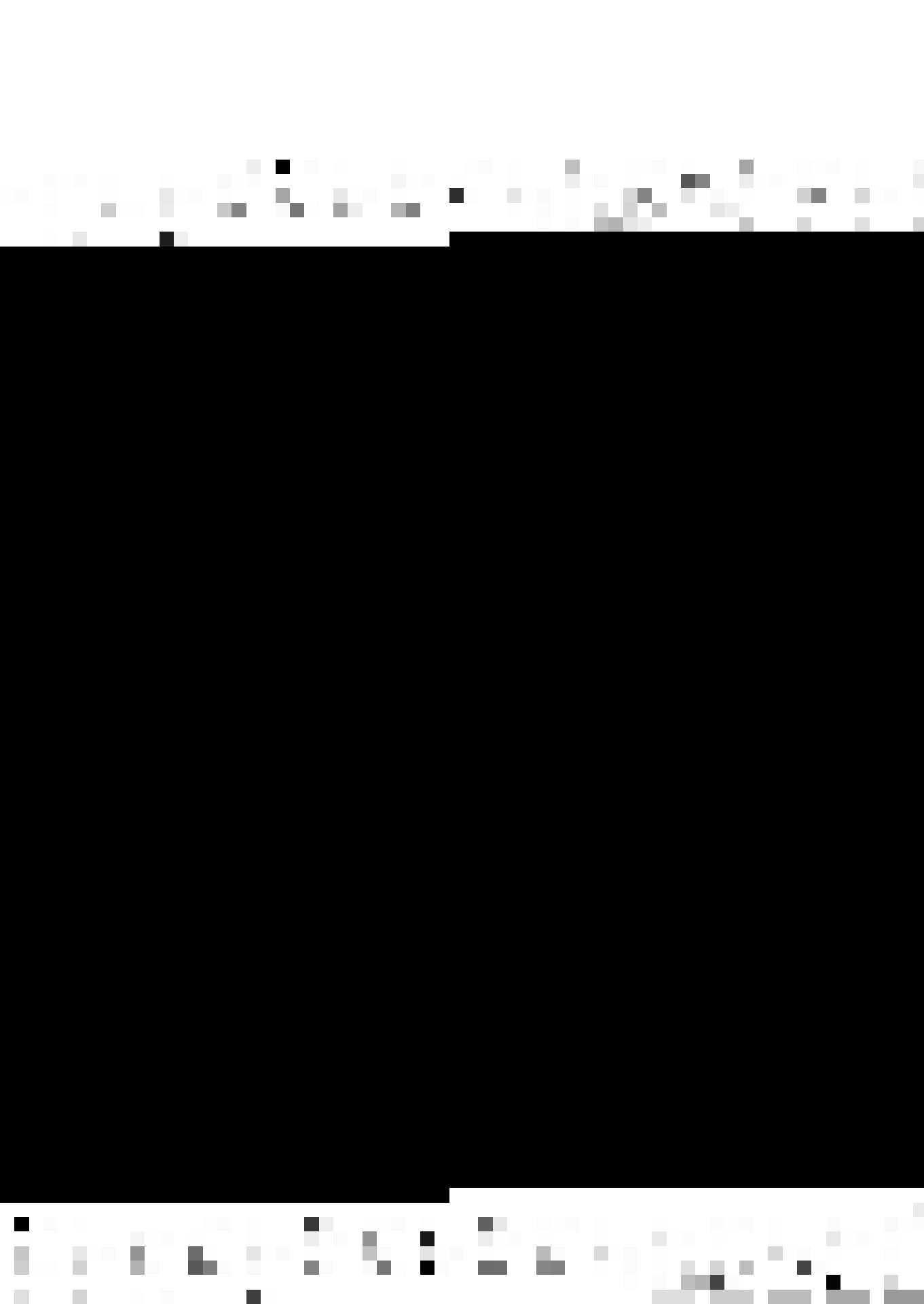


4.  $S_1 \rightarrow (CS_1)^n,$
5.  $S_1 \rightarrow (C)^n,$
6.  $S_2 \rightarrow (D_1)^n,$
7.  $S_2 \rightarrow (D_1 D_2)^n,$
8.  $S_2 \rightarrow (D_1 G_1)^n,$
9.  $G_i \rightarrow (D_{i+1} G_{i+1})^n, 1 \leq i \leq n'-4,$
10.  $G_{n'-3} \rightarrow (D_{n'-2} D_{n'-1})^n,$
11.  $G_i \rightarrow (D_{i+1} D_{i+2})^n, 1 \leq i \leq n'-4,$
12.  $C \rightarrow (D_1 F_1)^n,$
13.  $F_i \rightarrow (D_{i+1} F_{i+1})^n, 1 \leq i \leq n'-3,$
14.  $F_{n'-2} \rightarrow (D_{n'-1} D_{n'})^n,$
15.  $D_i \rightarrow (g'_{1i} A g_{1i}, \dots, g'_{ni} A g_{ni}), 1 \leq i \leq n',$
16.  $A \rightarrow (B_1 H_1)^n,$
17.  $H_i \rightarrow (B_{i+1} H_{i+1})^n, 1 \leq i \leq n-4,$
18.  $H_{n-3} \rightarrow (B_{n-2} B_{n-1})^n,$
19.  $B_i \rightarrow (B_{i1} B_{i2}, (\alpha_{i2}, \kappa_{i2}), \dots, (\alpha_{in}, \kappa_{in})), 1 \leq i \leq n-1$

where,

(i) if  $i = 1$  or  $i$  is in  $\xi_1$ , and

(a) if  $j$  is in  $\xi_1$ , then  $\alpha_{ij} = B_{i1} B_{i2}$  and  $\kappa_{ij}$  is an identity permutation, and



(b) if  $j$  in  $\{2, \dots, n\}$  is not in  $\xi_1$ , then

$\alpha_{ij} = B_{i2}B_{i1}$  and  $\pi'_{ij} = [2, 1]$ , and

(ii) if  $i$  in  $\{2, \dots, n-1\}$  is not in  $\xi_1$ , then locate the production:  $B \rightarrow (A_1A_2, (\beta_2, \pi'_2), \dots, (\beta_n, \pi'_n))$ , of the sample  $(n, \xi_1, \dots, \xi_{n-1}, 2)CG$  where  $\pi'_i$  is a non-identity permutation (notice that there is only one such production) and

(a) if  $\pi'_j$  is an identity production then  $\alpha_{ij} = B_{i1}B_{i2}$

and  $\pi'_{ij}$  is an identity permutation, and

(b) if  $\pi'_j$  is a non-identity permutation, then

$\alpha_{ij} = B_{i2}B_{i1}$  and  $\pi'_{ij} = [2, 1]$ ,

$$20. \quad B_{i1} \rightarrow (a_{i1}B_{i1}b_{i1})^n, \quad 1 \leq i \leq n-1,$$

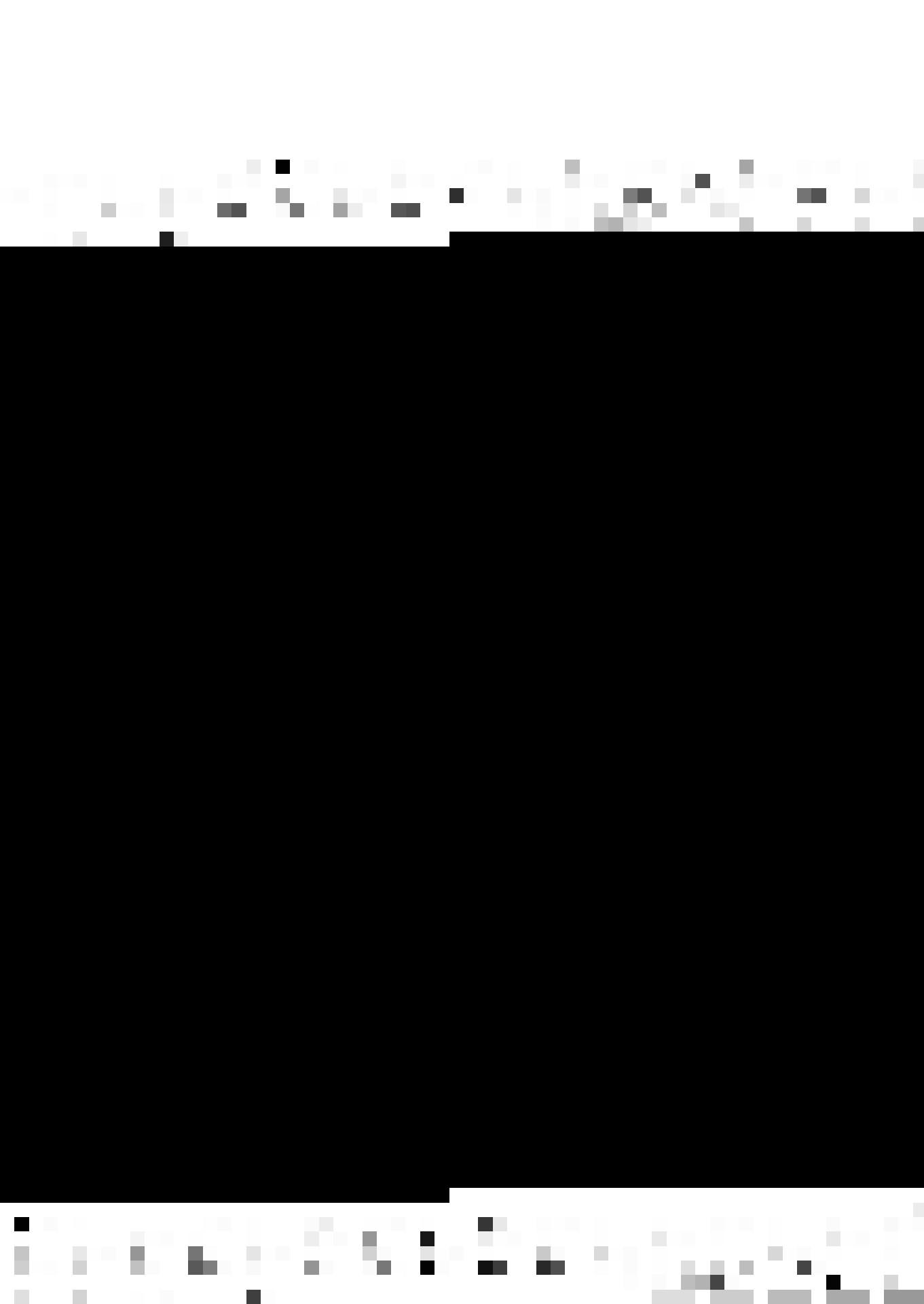
$$21. \quad B_{i1} \rightarrow (a_{i1}b_{i1})^n, \quad 1 \leq i \leq n-1,$$

$$22. \quad B_{i2} \rightarrow (a_{i2}B_{i2}b_{i2})^n, \quad 1 \leq i \leq n-1, \text{ and}$$

$$23. \quad B_{i2} \rightarrow (a_{i2}b_{i2})^n, \quad 1 \leq i \leq n-1.$$

Observe that  $G$  is an  $(n, \sigma, 2)CG$  and  $L(G)$  is the language

$$L_{(n, \xi_1, \dots, \xi_{n-1})}^{n'}$$



## APPENDIX D

This appendix supplements the proof of Lemma 2.4.2.

Here, we show that the language  $L_{(n, \xi_1, k)}^{n'}$  (defined in section 2.4) is in  $C_{(n, \sigma, k)}^L$ , where  $\sigma = \xi_1, \dots, \xi_{n-1}$ .

We shall construct an  $(n, \sigma, k)$ CG  $G$  such that  $G$  generates the language  $L_{(n, \xi_1, k)}^{n'}$ , by modifying the sample  $(n, \tau, k)$ CG (see the proof of Theorem 1.1.1).

Let  $G' = (V_N', \emptyset, P', (S_1)^{n'})$ , be the sample  $(n, \tau, k)$ CG.

$$G = (V_N, \{a_1, b_1, \dots, a_k, b_k, g_{11}, g'_{11}, \dots, g_{n1}, g'_{n1}, \dots, g_{1n'}, g'_{1n'}, \dots, g_{nn'}, g'_{nn'}, g_1, \dots, g_n\}, P, (S)^n),$$

where  $V_N$  and  $P$  are constructed as follows:

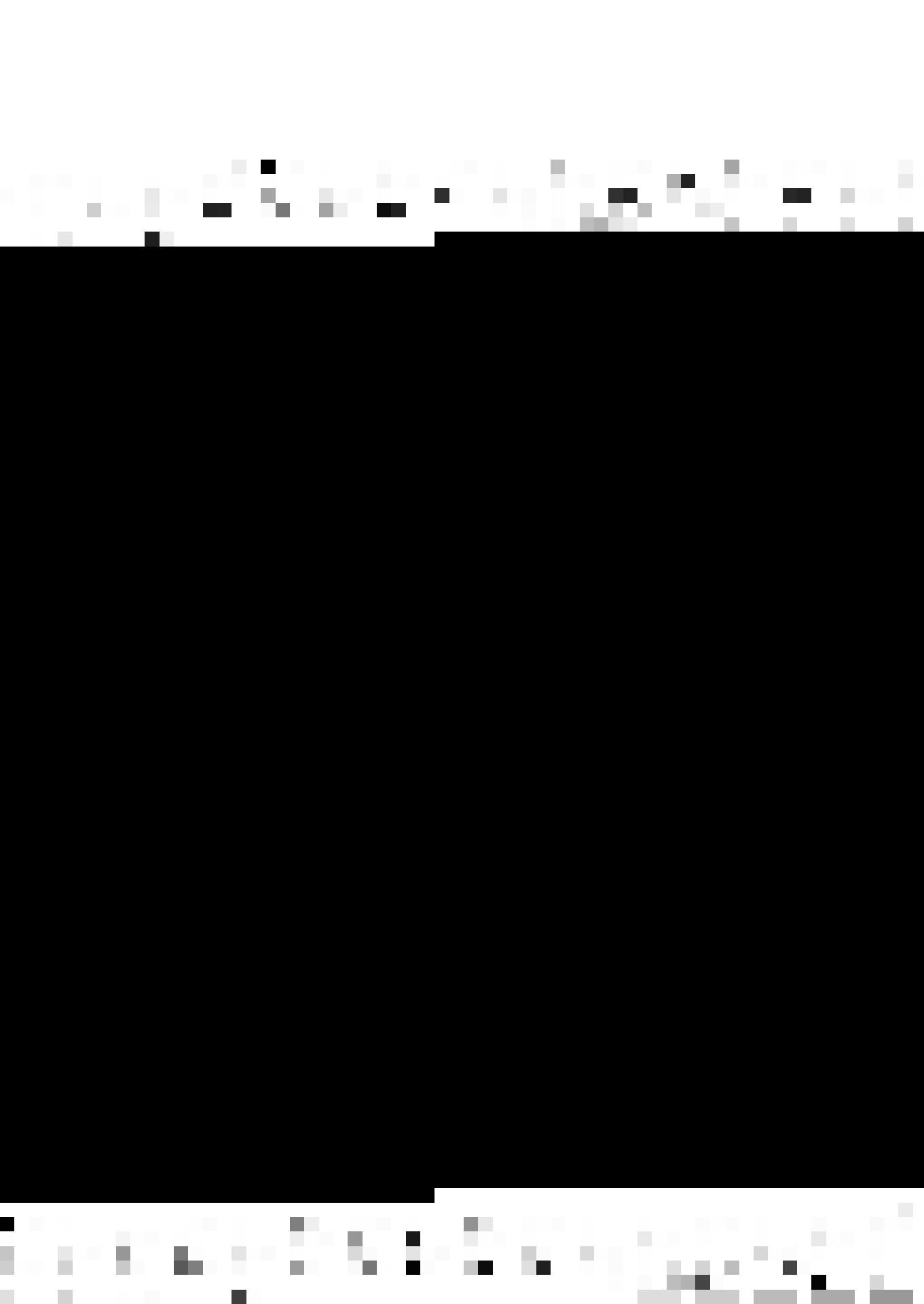
$$(a) \quad V_N = V_N' \cup V_N'', \text{ where}$$

$$\begin{aligned} V_N'' &= \{A, S, S', S_2, S_3, C\} \cup \\ &\quad \{G_i \mid 1 \leq i \leq n'-3\} \cup \\ &\quad \{F_i \mid 1 \leq i \leq n'-2\} \cup \\ &\quad \{D_i \mid 1 \leq i \leq n'\} \cup \{B_i \mid 1 \leq i \leq k\}. \end{aligned}$$

$$(b) \quad P = P', \text{ to start with.}$$

Add to  $P$  the production:

$A \rightarrow (B_1 \dots B_k, (\alpha_2, \pi_2), \dots, (\alpha_n, \pi_n))$ , where if  $i$  is in  $\xi_1$ , then  $\alpha_i = B_1 \dots B_k$ , and  $\pi_i$  is an identity permutation, and if  $i$  in  $\{2, \dots, n\}$  is not in  $\xi_1$ , then  $\alpha_i = B_{\pi_k(1)} \dots B_{\pi_k(k)}$  and  $\pi_i = \pi_k$ .



(c) Finally, add to P the following productions:

1.  $S \rightarrow (S'g_1, \dots, S'g_n),$
2.  $S' \rightarrow (S_2S_3)^n,$
3.  $S' \rightarrow (S_3)^n,$
4.  $S_2 \rightarrow (CS_2)^n,$
5.  $S_2 \rightarrow (C)^n,$
6.  $S_3 \rightarrow (D_1)^n,$
7.  $S_3 \rightarrow (D_1D_2)^n,$
8.  $S_3 \rightarrow (D_1G_1)^n,$
9.  $G_i \rightarrow (D_{i+1}G_{i+1})^n, 1 \leq i \leq n'-4,$
10.  $G_{n'-3} \rightarrow (D_{n'-2}D_{n'-1})^n,$
11.  $G_i \rightarrow (D_{i+1}D_{i+2})^n, 1 \leq i \leq n'-4,$
12.  $C \rightarrow (D_1F_1)^n,$
13.  $F_i \rightarrow (D_{i+1}F_{i+1})^n, 1 \leq i \leq n'-3,$
14.  $F_{n'-2} \rightarrow (D_{n'-1}D_{n'})^n,$
15.  $D_i \rightarrow (g'_1Ag_{1i}, \dots, g'_{ni}Ag_{ni}), 1 \leq i \leq n',$
16.  $B_i \rightarrow (a_iBb_i)^n, 1 \leq i \leq k,$
17.  $B_i \rightarrow (a_i b_i)^n, 1 \leq i \leq k.$

Observe that  $L(G)$  is the language  $L_{(n, \xi_1, k)}^{n'}.$



## APPENDIX E

In this appendix, we prove Lemma 3.1.1.

Lemma:

Let  $M = (Q, \Sigma, \Gamma, \delta_a, \delta_c, C, q_0, z_0, z_R, F)$ , be an nGCA. If for some  $q'$  in  $Q$ ,  $t \geq 1$ , and  $z_2, \dots, z_t$  in  $\Gamma - \{z_0, z_R\}$ ,

$$(q, w, \alpha z_1 \uparrow z \alpha', 0)$$

$$\vdash^{a^*} (q', w, \alpha z_1 \uparrow \dots \uparrow z_t z \uparrow \alpha', 0),$$

in  $x$  moves, where  $z_1, z$  are in  $\Gamma$ , then  $x = t$ .

Proof:

We prove this by induction on  $t$ .

Suppose  $t = 1$ . Then, clearly  $x = 1$ .

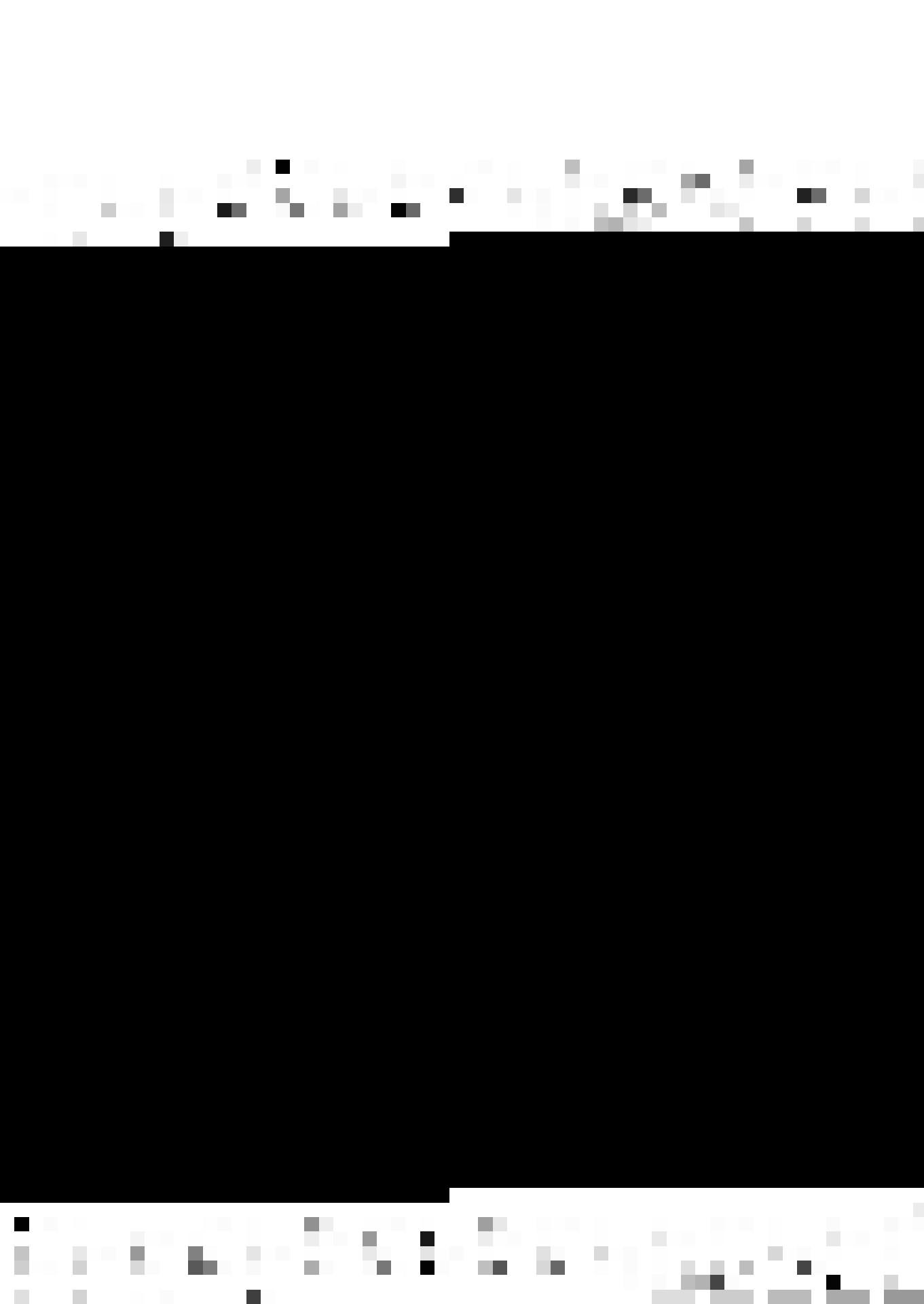
Now, suppose for  $t \leq k$ ,  $x = t$ . Assume now that  $t = k+1$ . Let us consider the first move. Because of this move, let

$$(q, w, \alpha z_1 \uparrow z \alpha', 0)$$

$$\vdash^{a} (q_1, w, \alpha z_1 z_{i_1} \uparrow z_{i_2} \dots z_{i_m} z \alpha', 0),$$

where  $m \geq 1$ . Clearly,  $2 = i_1 < \dots < i_m \leq t$ .

Using the induction hypothesis, we have:



$$(q_1, w, \alpha z_1 z_{i_1} \uparrow z_{i_2} \dots z_{i_m} z\alpha', 0)$$

$$\vdash^{a*} (q_2, w, \alpha z_1 z_{i_1} \dots z_{i_{2-1}} z_{i_2} \uparrow \dots z_{i_m} z\alpha', 0),$$

in  $i_2 - i_1$  moves,

•  
•  
•

$$(q_{m-1}, w, \alpha z_1 \dots z_{i_{m-1}} \uparrow z_{i_m} z\alpha', 0)$$

$$\vdash^{a*} (q_m, w, \alpha z_1 \dots z_{i_{m-1}} \dots z_{i_{m-1}} z_{i_m} \uparrow z\alpha', 0)$$

in  $i_m - i_{m-1}$  moves, and

$$(q_m, w, \alpha z_1 \dots z_{i_m} \uparrow z\alpha', 0)$$

$$\vdash^{a*} (q', w, \alpha z_1 \dots z_{i_m} \dots z_t z \uparrow \alpha', 0),$$

in  $t - i_m + 1$  moves.

Thus,

$$(q, w, \alpha z_1 \uparrow z\alpha', 0),$$

$$\vdash^{a*} (q', w, \alpha z_1 \dots z_t z \uparrow \alpha', 0),$$

in  $1 + i_2 - i_1 + i_3 - i_2 + \dots + i_m - i_{m-1} + t - i_m + 1$ , moves i.e. in

$t+1+i_1-i_1 = t+2-2 = t$  moves.

Q.E.D.



## APPENDIX F

In this appendix, we prove statement (2) made in the proof of Theorem 3.2.1. It should be read in conjunction with that proof.

Lemma:

$i_1, \dots, i_j$  is a terminating sequence of productions in  $G'$  iff

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash_M^{a*} (q_0, w, y' z_{i_1} \dots z_{i_j} z \uparrow y, 0), \text{ where}$$

$j \geq 1$  and  $z_{i_1}, z$  are in  $P$ .

Proof:

Let  $A$  be the left hand side of the production  $i_1$  and let  $(A)^n \xrightarrow[G]{*} (x_1, \dots, x_n)$  by a leftmost derivation using the production sequence  $i_1, \dots, i_j$ .

only if

Let  $j = 1$ . This implies that there is a production:  $A \rightarrow (x_1, \dots, x_n)$  in  $P'$  and from the construction of  $M$ , we then have:  $s_a(q_0, z_{i_1}) = \{(q_0, \epsilon, 0)\}$ , and therefore,

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash_M^{a*} (q_0, w, y' z_{i_1} z \uparrow y, 0)$$



Again, suppose as an induction hypothesis that the 'only if' part of the lemma is true for  $j \leq k$ .

Now, assume that  $j = k+1$ . Either (i) or (ii) can be the case:

(i)  $i_1$ , the first production used in the derivation is the nonterminating production:  $A \rightarrow (B)^n$ .

We observe that in the derivation, B will be expanded using the production  $i_2$ . From the construction of  $M$ , we then have:

$$\delta_a(q_0, z_{i_1}) = \{(q_0, z_{i_2}, 0)\}, \text{ so that}$$

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash^a (q_0, w, y' z_{i_1} z_{i_2} \uparrow zy, 0)$$

Now, from the induction hypothesis, it follows that

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash^{a*} (q_0, w, y' z_{i_1} \dots z_{i_k} z \uparrow y, 0)$$

(ii)  $i_1$ , the first production used in the derivation is the nonterminating production:  $A \rightarrow (BC)^n$ , for some  $B, C$  in  $V_N^*$ .

We observe that in the derivation, B will be expanded using the production  $i_2$ . Let  $C$  be expanded using some production  $i_s$ . This implies that  $(B)^n \xrightarrow[G]{*} (x'_1, \dots, x'_n)$ ,  $x'_i, 1 \leq i \leq n$ , in  $V_T^*$ , by a leftmost derivation, using the production sequence  $i_2, \dots, i_{s-1}$ ,  $(C)^n \xrightarrow[G']{*} (x''_1, \dots, x''_n)$ ,  $x''_i, 1 \leq i \leq n$ , in  $V_T^*$  by a leftmost derivation using the



production sequence  $i_s, \dots, i_j$ , where  $x'_i = x''_i = x_i$ ,  
 $1 \leq i \leq n$ .

From the construction of  $M$ , we then have:

$$\delta_a(q_0, z_{i_1}) = \{(q_0, z_{i_2} z_{i_s}, 0)\}, \text{ so that}$$

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash^a (q_0, w, y' z_{i_1} z_{i_2} \uparrow z_{i_s} zy, 0).$$

Now, from the induction hypothesis, the 'only if' part of the lemma follows.

if

From Lemma 3.1.1, we have

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\vdash^{a*} (q_0, w, y' z_{i_1} \dots z_{i_j} z \uparrow y, 0), \text{ in } j \text{ moves.}$$

We will prove this part of the lemma by induction on  $j$  (which is equal to the number of moves required, from above).

Suppose  $j = 1$ . In that case, clearly  $\delta_a(q_0, z_{i_1}) = \{(q_0, e, 0)\}$  and so  $i_1$  must be a terminating production in  $P$  (follows from the construction of  $M$ ), so that  $i_1$  is a terminating sequence of productions.

Again, suppose as an induction hypothesis that the 'if' part of the lemma is true for  $j \leq k$ .



Now, assume that  $j = k+1$ . Either (iii) or (iv) can be the case.

(iii) In the first move, the rule used is:

$$\delta_a(q_0, z_{i_1}) = \{(q_0, z_{i_2} z_{i_1}, 0)\}, \text{ so that}$$

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\xrightarrow{a} (q_0, w, y' z_{i_1} z_{i_2} \uparrow z_{i_1} zy, 0).$$

From the construction of  $M$ , we observe that the production  $i_1$  in  $P'$  must be:  $A \rightarrow (BC)^n$ , such that the production  $i_2$  and  $i_1$  have  $B$  and  $C$  as their left hand sides, respectively.

Clearly, we can write,

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\xrightarrow{a^*} (q_0, w, y' z_{i_1} \dots z_{i_j} z \uparrow y, 0)$$

as

$$(q_0, w, y' z_{i_1} \uparrow zy, 0)$$

$$\xrightarrow{a} (q_0, w, y' z_{i_1} z_{i_2} \uparrow z_{i_1} zy, 0)$$

$$\xrightarrow{a^*} (q_0, w, y' z_{i_1} \dots z_{i_l} \uparrow zy, 0)$$

$$\xrightarrow{a^*} (q_0, w, y' z_{i_1} \dots z_{i_l} \dots z_{i_j} z \uparrow y, 0).$$

The 'if' part of the lemma follows now using the induction hypothesis.



(iv) In the first move, the rule used is:

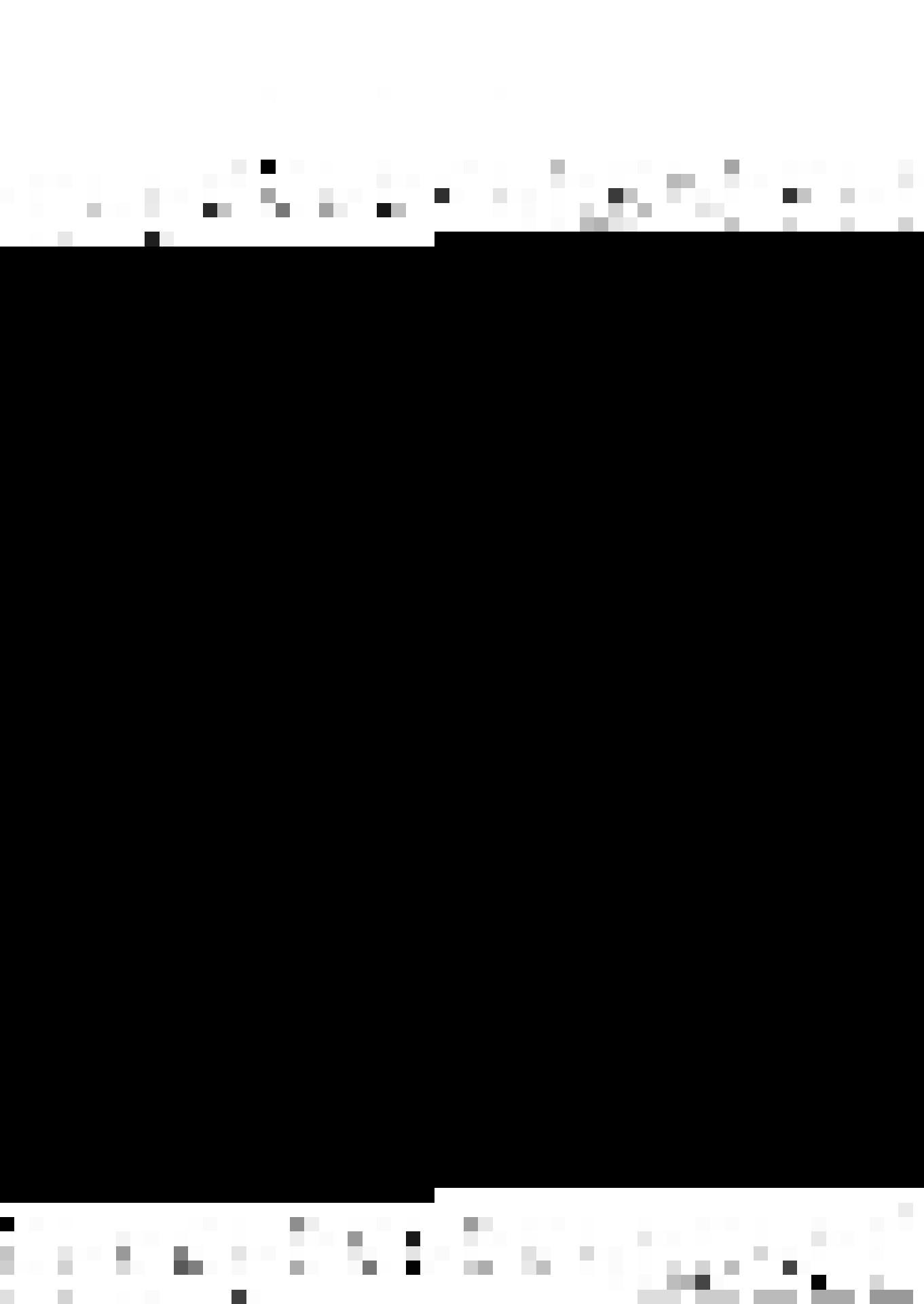
$$\delta_a(q_0, z_{i_1}) = \{(q_0, z_{i_2}, o)\}, \text{ so that}$$

$$(q_0, w, y' z_{i_1} \uparrow zy, o)$$

$$\vdash^a (q_0, w, y' z_{i_1} z_{i_2} \uparrow y, o).$$

In this case, the proof is similar to that for case (iii), and is omitted.

Q.E.D.



## APPENDIX G

In this appendix, we shall essentially establish the equivalence between the class of  $(1, n, k)$ -coupled translations,  $k \geq 2$ , and the class of translations defined by  $k$ -register  $n$ -pushdown assemblers (see section 3.3). This we shall do by proving Theorem 1 and Theorem 2.

### Theorem 1:

If a translation  $T$  is a  $(1, n, k)$ -coupled translation, then  $T = T(M)$  for some  $k'$ -register  $n$ -pushdown assembler  $M$ , where  $k' = k$  if  $k \geq 2$ , and  $k' \leq 2$ , otherwise.

### Proof:

Let  $T = T_1(G)$ , where

$$G = (V_N, \Sigma \cup \Delta, P, (S)^{n+1}), \quad \Sigma = V_{T_1},$$

$\Delta = V_{T_2} \cup \dots \cup V_{T_{n+1}}$ , is an  $(n+1, k')$ -CG in normal form (see Theorem 1.4.1).

Let  $M = (Q, \Sigma, \Delta, \Gamma, \lambda, \mu, \delta, q_0, S)$ , where the various parameters of the machine are defined as follows:

$Q$  consists of the symbol  $q_0$  and the finite set of symbols of the form  $q_a$ , where  $a$  is in  $\Sigma \cup \{\epsilon\}$  and there is a production in  $P$ :

$$A \rightarrow (a, b_1, \dots, b_n).$$



$\Gamma'$  consists of:

(a) Symbols in  $V_N$ ,

(b) Symbols  $[\alpha, \bar{\kappa}_1, \bar{\kappa}_2, \dots, \bar{\kappa}_n]$ , where  $\alpha$  is in  $V_N^*$ , there is a production:  $A \rightarrow (\gamma, (\beta_1, \bar{\kappa}_1), \dots, (\beta_n, \bar{\kappa}_n))$ , in  $P$ , and  $\alpha$  is a suffix of  $\gamma$ .

(c) Symbols  $[a]$ , where  $a$  is in  $\Sigma \cup \{\epsilon\}$ .

We define  $\lambda$ ,  $\mu$ ,  $\nu$  by:

1. Suppose  $A \rightarrow (\alpha, (\beta_1, \bar{\kappa}_1), \dots, (\beta_n, \bar{\kappa}_n))$ , is a production in  $P$ .

If  $\alpha$  is in  $V_N^* - \{\epsilon\}$ , then

(i)  $\lambda(q_0, \epsilon, A)$  contains  $(q_0, [\alpha, \bar{\kappa}_1, \bar{\kappa}_2, \dots, \bar{\kappa}_n])$ .

If  $\alpha$  is in  $\Sigma \cup \{\epsilon\}$ , then

(ii)  $\mu(q_0, \epsilon, A)$  contains  $(q_\alpha, (\beta_1, 1), (\beta_2, 1), \dots, (\beta_n, 1))$ ,

(iii)  $\lambda(q_\alpha, \epsilon, A)$  contains  $(q_0, [\alpha])$ .

2. (i)  $\lambda(q_0, a, [a]) = \{(q_0, [\epsilon])\}$ , for all  $a$  in  $\Sigma$ ,

(ii)  $\lambda(q_0, \epsilon, [\epsilon]) = \{(q_0, \epsilon)\}$ .

3. (i)  $\lambda(q_0, \epsilon, [A\alpha, \bar{\kappa}_1, \bar{\kappa}_2, \dots, \bar{\kappa}_n])$   
 $= \{(q_0, A[\alpha, \bar{\kappa}_1, \bar{\kappa}_2, \dots, \bar{\kappa}_n])\}$

(ii)  $\nu(q_0, \epsilon, [\alpha, \bar{\kappa}_1, \bar{\kappa}_2, \dots, \bar{\kappa}_n])$   
 $= \{(q_0, i_1, \dots, i_n)\}, i_j, 1 \leq j \leq n,$



is equal to  $\pi_j(m - |\alpha|)$  if  $\pi_1$  is a permutation on  $m$  objects.  
 $(|\alpha|$  denotes the number of symbols in the string  $\alpha$ ).

$$(iii) \lambda(q_0, \epsilon, [\epsilon, \pi_1, \pi_2, \dots, \pi_n]) = \{(q_0, \epsilon)\}.$$

We shall now prove that

$$(q_0, w, A(t_0)^n, T_0) \vdash_M^* (q_0, \epsilon, \emptyset, (x_1, \dots, x_n)),$$

where  $t_0 = (\emptyset)^m$  and  $T_0 = (\emptyset)^n$

iff

$$(A)^{n+1} \xrightarrow[G]{*} (w, x_1, \dots, x_n),$$

for any nonterminal  $A$ ,  $w$  in  $\Sigma^*$  and  $x_i$ ,  $1 \leq i \leq n$ , in  $\Delta^*$ .

From this result, it will immediately follow that

$$T(M) = T_1(G).$$

if

We will prove the result by induction on the number of steps used in the derivation:

$$(A)^{n+1} \xrightarrow[G]{*} \dots \xrightarrow[G]{*} (w, x_1, \dots, x_n).$$

The one step case follows directly from an application of rules 1(ii), 1(iii), 2(i) (zero number of times or just once), and 2(ii).

Assume the result for less than  $j$  steps,  $j \geq 2$ , then the first step in a  $j$  step derivation must be of the type:



$$(A)^{n+1} \xrightarrow[G]{\quad} (A_1 \dots A_m, (A \hat{\wedge}_1(1) \dots A \hat{\wedge}_1(m), \bar{\kappa}_1), \dots, \dots, (A \hat{\wedge}_n(1) \dots A \hat{\wedge}_n(m), \bar{\kappa}_n)).$$

By rule 1(i), we know that

$$(q_0, w, A(t_0)^n, t_0)$$

$$\vdash_M^* (q_0, w, [A_1 \dots A_m, \bar{\kappa}_1, \dots, \bar{\kappa}_n] (t_0)^n, t_0).$$

We can write  $w = w_1 \dots w_m$  and

$x_i$ ,  $1 \leq i \leq n$ , as  $x_i = x_{i1} \dots x_{im}$ , such that

$$(A_i)^{n+1} \xrightarrow[G]{\quad} (w_i, {}^x \bar{\kappa}_1(i), {}^x \bar{\kappa}_2(i), \dots, {}^x \bar{\kappa}_n(i)),$$

for each  $i$ .

Thus, by the induction hypothesis,

$$(\#): (q_0, w_i, A_i(t_0)^n, t_0)$$

$$\vdash_M^* (q_0, \epsilon, \emptyset, ({}^x \bar{\kappa}_1(i), \dots, {}^x \bar{\kappa}_n(i))).$$

Using rule 3(i), relation  $(\#)$  and rule 3(ii),  $m$  times, then rule 3(iii), one can easily put together a sequence of moves of  $M$  which demonstrate that

$$(q_0, w, z_0(t_0)^n, t_0)$$

$$\vdash_M^* (q_0, \epsilon, \emptyset, (x_1, \dots, x_n))$$

only if

We will prove by induction on the number of moves made by  $M$  that if



$$(q_0, w, A(t_0)^n, T_0)$$

$$\overline{M}^* (q_0, \epsilon, \emptyset, (x_1, \dots, x_n));$$

then

$$(A)^{n+1} \xrightarrow[G]{*} (w, x_1, \dots, x_n).$$

The result is true vacuously for fewer than three moves. Suppose it is true for less than  $j$  moves. For a sequence of  $j$  moves, the first move must be due either to rule 1(i) or 1(ii). In the latter case, the subsequent moves of  $M$  are completely determined by rules 1(iii), 2(i) and 2(ii). Moreover, by 1(ii) there is a production:  $A \rightarrow (w, x_1, \dots, x_n)$ , in  $P$ . The result we desire follows in this case without reference to the hypothesis.

If the first move is due to 1(i), we can express the subsequent operation of  $M$  as:

$$(q_0, w_1 \dots w_m, [A_1 \dots A_m, \bar{x}_1, \dots, \bar{x}_n](t_{01}, \dots, t_{0n}), T_0)$$

$$\overline{M} (q_0, w_1 \dots w_m, A_1(t_{01}, \dots, t_{0n})[A_2 \dots A_m,$$

$$\bar{x}_1, \dots, \bar{x}_n](t_{01}, \dots, t_{0n}), T_0),$$

$$(q_0, w_1 \dots w_m, A_1(t_{01}, \dots, t_{0n})[A_2 \dots A_m, \bar{x}_1, \dots$$

$$\dots, \bar{x}_n](t_{01}, \dots, t_{0n}), T_0)$$

$$\overline{M}^* (q_0, w_2 \dots w_m, [A_2 \dots A_m, \bar{x}_1, \dots, \bar{x}_n](t_{01}, \dots$$

$$\dots, t_{0n}), (x_{11}, \dots, x_{1n})),$$



- $(q_0, w_2 \dots w_m, [A_2 \dots A_m, \kappa_1, \dots, \kappa_n](t_{01}, \dots, t_{0n}), (x_{11}, \dots, x_{1n}))$   
 $\vdash_M (q_0, w_2 \dots w_m, [A_2 \dots A_m, \kappa_1, \dots, \kappa_n](t_{11}, \dots, t_{1n}), T_0),$   
 $(q_0, w_2 \dots w_n, [A_2 \dots A_m, \kappa_1, \dots, \kappa_n](t_{11}, \dots, t_{1n}), T_0)$   
 $\vdash_M (q_0, w_2 \dots w_m, A_2(t_{01}, \dots, t_{0n})[A_3 \dots$   
 $\dots A_m, \kappa_1, \dots, \kappa_n](t_{11}, \dots, t_{1n}), T_0),$   
 $\vdots$   
 $\vdash_M (q_0, w_m, A_m(t_{01}, \dots, t_{0n})[\epsilon, \kappa_1, \dots, \kappa_n](t_{(m-1)1}, \dots, t_{(m-1)n}), T_0)$   
 $\vdash_M^*(q_0, \epsilon, [\epsilon, \kappa_1, \dots, \kappa_n](t_{(m-1)1}, \dots, t_{(m-1)n}), (x_{m1}, \dots, x_{mn})),$   
 $(q_0, \epsilon, [\epsilon, \kappa_1, \dots, \kappa_n](t_{(m-1)1}, \dots, t_{(m-1)n}), (x_{m1}, \dots, x_{mn}))$   
 $\vdash_M (q_0, \epsilon, [\epsilon, \kappa_1, \dots, \kappa_n](t_{m1}, \dots, t_{mn}), T_0),$   
 $(q_0, \epsilon, [\epsilon, \kappa_1, \dots, \kappa_n](t_{m1}, \dots, t_{mn}), T_0)$   
 $\vdash_M (q_0, \epsilon, \emptyset, (x_1, \dots, x_n));$



here  $t_{0i} = (\emptyset)^k$  and  $t_{si}$  is  $t_{(s-1)i}$  with  $x_{si}$  replacing  $\emptyset$  in the  $\pi_i(s)$ -th position,  $1 \leq i \leq n$ ,  $1 \leq s \leq m$ .

The initial move of  $M$  implies that  $P$  has a production:

$$A \rightarrow (A_1 \dots A_m, A \hat{\pi}_1(1) \dots A \hat{\pi}_1(m), \dots, A \hat{\pi}_n(1) \dots A \hat{\pi}_n(m)).$$

By the induction hypothesis, the above sequence of moves of  $M$  implies that

$$(A_i)^{n+1} \xrightarrow[G]{*} (w_i, x_{i1}, \dots, x_{in}),$$

for all  $i$ ,  $1 \leq i \leq m$ .

Putting the above together we can easily show that

$$(A)^{n+1} \xrightarrow[G]{*} (w, x_1, \dots, x_n)$$

Q.E.D.

To prove the converse of the above theorem viz.  
Theorem 2, we will need two auxiliary lemmas.

#### Lemma 1:

Every translation defined by some  $k$ -register  $n$ -pushdown assembler  $M'$ , is defined by a  $k$ -register  $n$ -pushdown assembler,

$M = (Q, \Sigma, \Delta, \Gamma, \lambda, \mu, \nu, q_0, z_0)$ , for which if  $\lambda(q, a, z)$  contains  $(p, \alpha)$ , then  $|\alpha| \leq 2$ .



Proof:

Let  $M' = (Q', \Sigma, \Delta, \Gamma, \lambda', U, D, q_0, z_0)$ .

For each  $(p, z_1 \dots z_m)$  in  $\lambda'(q, a, z)$  with  $m > 2$ , introduce new states  $q_1, q_2, \dots, q_{m-2}$  to  $Q$ . Remove  $(p, z_1 \dots z_m)$  from  $\lambda'(q, a, z)$  and replace it by  $(q_1, z_{m-1} z_m)$ .

Define

$$\lambda(q_i, \epsilon, z_{m-i}) = \{(q_{i+1}, z_{m-i-1} z_{m-i})\},$$

for  $i = 1, 2, \dots, m-3$ , and

$$\lambda(q_{m-2}, \epsilon, z_2) = \{(p, z_1 z_2)\}.$$

After making all such replacements,  $Q$  is the resulting set of states and  $\lambda$  the result of making these alterations in  $\lambda'$ .

The validity of the result is now easy to see.

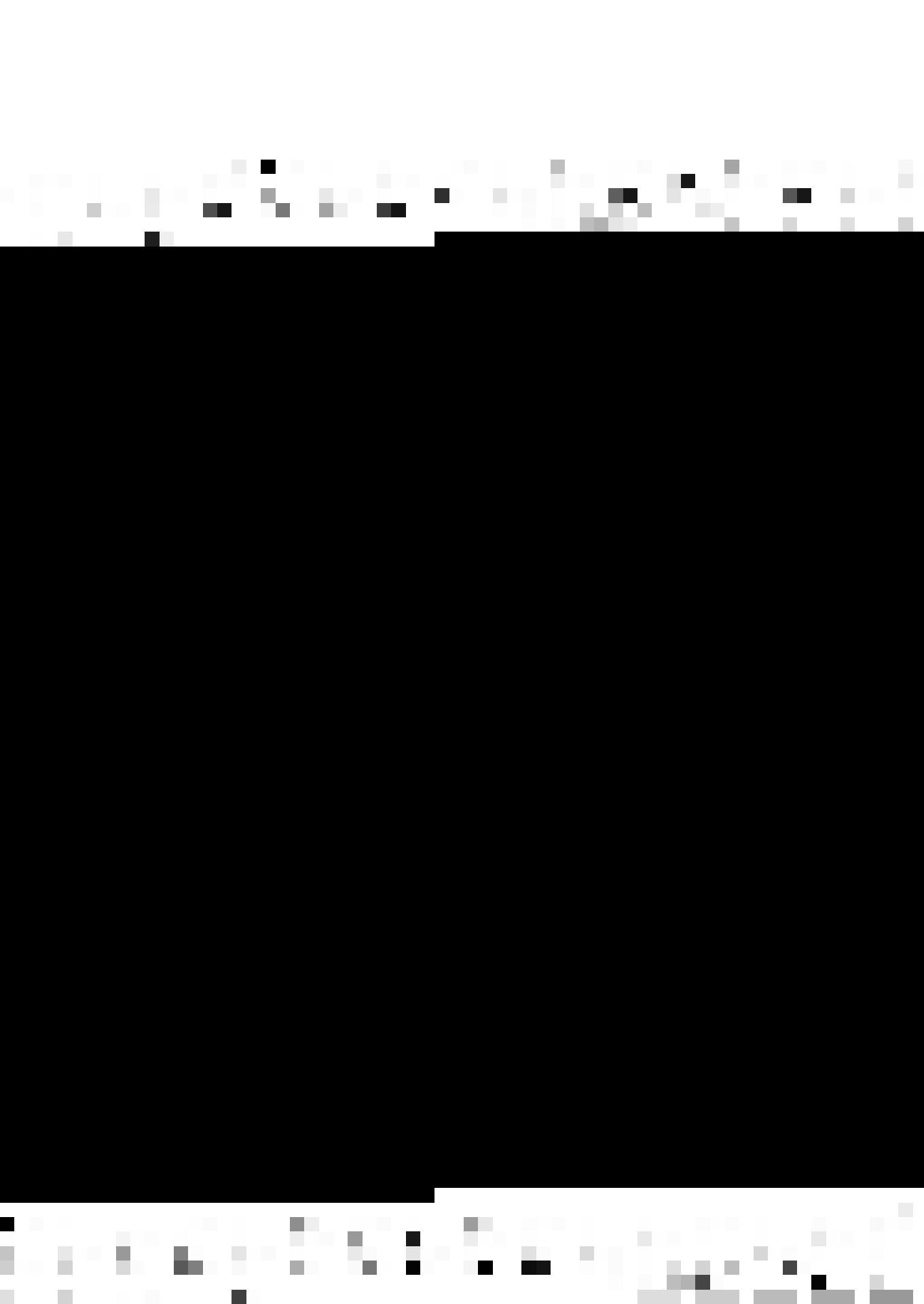
Q.E.D.

Lemma 2:

Every translation defined by a k-register n-pushdown assembler  $M'$ , is defined by a k-register n-pushdown assembler  $M$ , which satisfies Lemma 1 and has the additional property that if it erases an entry in its pushdown store, then that entry has no empty registers associated with it.

Proof:

It will be observed that for a k-register n-pushdown assembler, at any stage of computation, at any level, the



number of registers which are filled in each set at the level will be the same. This is because the only mappings which cause the registers to be filled are  $\lambda$  and  $\lambda'$ , and on any move one register from each set at the top level is filled provided all these registers are empty.

Now, let  $M' = (Q', \Sigma, \Delta, \Gamma', \lambda', \mu', \nu', q_0, z_0)$ , and assume that  $M'$  satisfies Lemma 1.  $M$  will simulate  $M'$ , but in addition, in the control symbol of each entry in the pushdown store,  $M$  will keep track of which registers in each of the  $n$  sets of registers associated with that entry, are empty. If  $M'$  erases an entry,  $M$  first stores  $\epsilon$  in each empty register.

Formally, let  $M = (Q_1, \Sigma, \Delta, \Gamma, \lambda, \mu, \nu, q_0, [z_0, (\emptyset)^n])$ .

Let  $K = \{1, \dots, k\}$  and

$\Gamma = \{[x, (s_1, \dots, s_n)] \mid x \text{ is in } \Gamma' \text{ and } s_i, 1 \leq i \leq n, \text{ is a subset of } K\}$ .

$Q$  contains the states of  $Q'$  and some new states which are introduced through the definitions of  $\lambda$ ,  $\mu$ , and  $\nu$ , below.

For all  $q$  and  $p$  in  $Q'$ ,  $a$  in  $\Sigma \cup \{\epsilon\}$ ,  $x$ ,  $y$  and  $z$  in  $\Gamma'$ ,  $s_i$ ,  $1 \leq i \leq n$ , subset of  $K$ :

1. If  $\lambda'(q, a, z)$  contains  $(p, y)$  then  $\lambda(q, a, [z, (s_1, \dots, s_n)])$  contains  $(p, [y, (s_1, \dots, s_n)])$ . If  $\lambda'(q, a_1, z)$  contains  $(p, xy)$  then  $\lambda(q, a, [z, (s_1, \dots, s_n)])$  contains  $(p, [x, (\emptyset)^n] [y, (s_1, \dots, s_n)])$ .



(M keeps track of full registers at each level when manipulating the pushdown store).

2. If  $\lambda'(q, a, z)$  contains  $(p, \epsilon)$  and  $s_j, 1 \leq j \leq n$  let  $\{i_{1j}, \dots, i_{mj}\}$  be  $K - s_j$ . For this  $(s_1, \dots, s_n)$ , introduce to Q new states  $q_1, \dots, q_m$ . Let

$\mu(q, a, [z, (s_1, \dots, s_n)])$  contain

$(q_1, (\epsilon, i_{11}), \dots, (\epsilon, i_{1n})),$

$(q_1, \epsilon, [z, (s_1, \dots, s_n)])$  contain

$(q_2, (\epsilon, i_{21}), \dots, (\epsilon, i_{2n})),$

$\vdots$

$\vdots$

$\vdots$

$\mu(q_{m-1}, \epsilon, [z, (s_1, \dots, s_n)])$  contain

$(q_m, (\epsilon, i_{m1}), \dots, (\epsilon, i_{mn})),$  and

$\lambda(q_m, \epsilon, [z, (s_1, \dots, s_n)])$  contain  $(p, \epsilon)$ .

If each  $s_j, 1 \leq j \leq n, = K$ , let

$\lambda(q, \epsilon, [z, (s_1, \dots, s_n)]),$  contain  $(p, \epsilon)$ .

(If M' would erase an entry that has empty registers, these registers are filled with  $\epsilon$  before erasing).

3. If  $i_j$  is not in  $S_j, 1 \leq j \leq n$ , and  $\lambda(q, a, z)$  contains  $(p, (w_1, i_1), \dots, (w_n, i_n))$ , introduce to Q a new state  $q_1$ .



Let  $\lambda(q, a, [z, (s_1, \dots, s_n)])$  contain

$(q_1, [z, (s_1 \cup \{i_1\}, \dots, s_n \cup \{i_n\})])$ , and

$\mu(q_1, \epsilon, [z, (s_1 \cup \{i_1\}, \dots, s_n \cup \{i_n\})])$  contain

$(p, (w_1, i_1), \dots, (w_n, i_n))$ .

(If  $M'$  stores output strings,  $M$  updates the set of full registers).

4. If  $i_j$  is not in  $s_j$ ,  $1 \leq j \leq n$ , and  $\nu(q, a, z)$  contains  $(p, i_1, \dots, i_n)$ , introduce to  $Q$  a new state  $q_1$ . Let  $\nu(q, a, [z, (s_1, \dots, s_n)])$  contain  $(q_1, i_1, \dots, i_n)$  and  $\lambda(q_1, \epsilon, [z, (s_1, \dots, s_n)])$  contain  $(p, [z, (s_1 \cup \{i_1\}, \dots, s_n \cup \{i_n\})])$ .

(When  $M'$  transfers the contents of the waiting registers i.e. the result of the erasure of the entry above,  $M'$  also updates the set of full registers).

Q.E.D.

### Theorem 2:

If a translation  $T = T(M)$  for a  $k$ -register  $n$ -pushdown assembler  $M$  then  $T = T_1(G)$  for an  $(n+1, k)CG G$ .

### Proof:

Let  $M = (Q, \Sigma, \Delta, \Gamma, \lambda, \mu, \nu, q_0, z_0)$ .

Assume that  $M$  satisfies Lemma 1 and Lemma 2.



$$G = (V_N, V_{T_1} \cup \dots \cup V_{T_{n+1}}, P, (S)^{n+1}),$$

where  $V_{T_1} = \Sigma$ ,  $V_{T_2} \cup \dots \cup V_{T_{n+1}} = \Delta$ ,  $V_N$  consists of:

1. The symbol  $S$ ,

2. a symbol  $[q, z, p]$  for each  $q$  and  $p$  in  $Q$  and  $Z$  in  $\Gamma$ , and

3. a symbol  $\langle q, Y, p, Z, i_1, \dots, i_n \rangle$  for each  $q$  and  $p$  in  $Q$ ,  $Y$  and  $Z$  in  $\Gamma$ ,  $i_j$ ,  $1 \leq j \leq n$ ,  $\leq k$ ,

and  $P$  is as specified below:

The idea is to let the symbols  $[q, z, p]$  and  $\langle q, Y, p, Z, i_1, \dots, i_n \rangle$  contain the following information:

For  $x$  in  $\Sigma^*$  and  $y_j$ ,  $1 \leq j \leq n$ , in  $\Delta^*$ ,

$$([q, z, p])^{n+1} \xrightarrow[G]{*} (x, y_1, \dots, y_n)$$

iff

$$(q, x, z(t_0)^n, t_0)$$

$$\vdash_M^* (p, \epsilon, \emptyset, (y_1, \dots, y_n)), \text{ and}$$

$$(\langle q, Y, p, Z, i_1, \dots, i_n \rangle)^{n+1}$$

$$\xrightarrow[G]{*} (x, y_1, \dots, y_n)$$

iff

$$(q, x, z(t_1, \dots, t_n), t_0)$$

$$\vdash_M^* (p, \epsilon, Y(t'_1, \dots, t'_n), (y_1, \dots, y_n)),$$



where  $t_0 = (\emptyset)^k$ ,  $T_0 = (\emptyset)^n$ , the  $i_j$ -th component of  $t_j$ ,  $1 \leq j \leq n$ , is  $\emptyset$  and  $t'_j$  is  $t_j$  with  $y_j$  in the  $j$ -th register of the set of registers represented by  $t_j$ .

The productions in  $P$  are:

$$1. \quad S \rightarrow ([q_0, z_0, p])^{n+1}, \text{ for each } p \text{ in } Q.$$

(This is to ensure that  $(S)^{n+1} \xrightarrow[G]{*} (x, y_1, \dots, y_n)$  such that  $(q_0, x, z_0(t_0)^n, T_0) \xrightarrow[M]{*} (p, \epsilon, \emptyset, (y_1, \dots, y_n))$ ).

2. Suppose  $\pi_1, \dots, \pi_n$  are permutations on  $k$  objects.

If for  $q_1, \dots, q_{k+1}$ ,  $p$  in  $Q$ ,  $z_1, \dots, z_{k+1}$  in  $\Gamma$ , and  $a$  in  $\Sigma \cup \{\epsilon\}$ ,  $\lambda(q_{k+1}, a, z_{k+1})$  contains  $(p, \epsilon)$ , then

$$\begin{aligned} [q_1, z_1, p] &\rightarrow (A_1 \dots A_k a, (A \hat{\pi}_1(1) \dots A \hat{\pi}_1(k), \pi_1), \dots \\ &\quad \dots, (A \hat{\pi}_n(1) \dots A \hat{\pi}_n(k), \pi_n)), \end{aligned}$$

where  $A_i$ ,  $1 \leq i \leq k$ , is the symbol

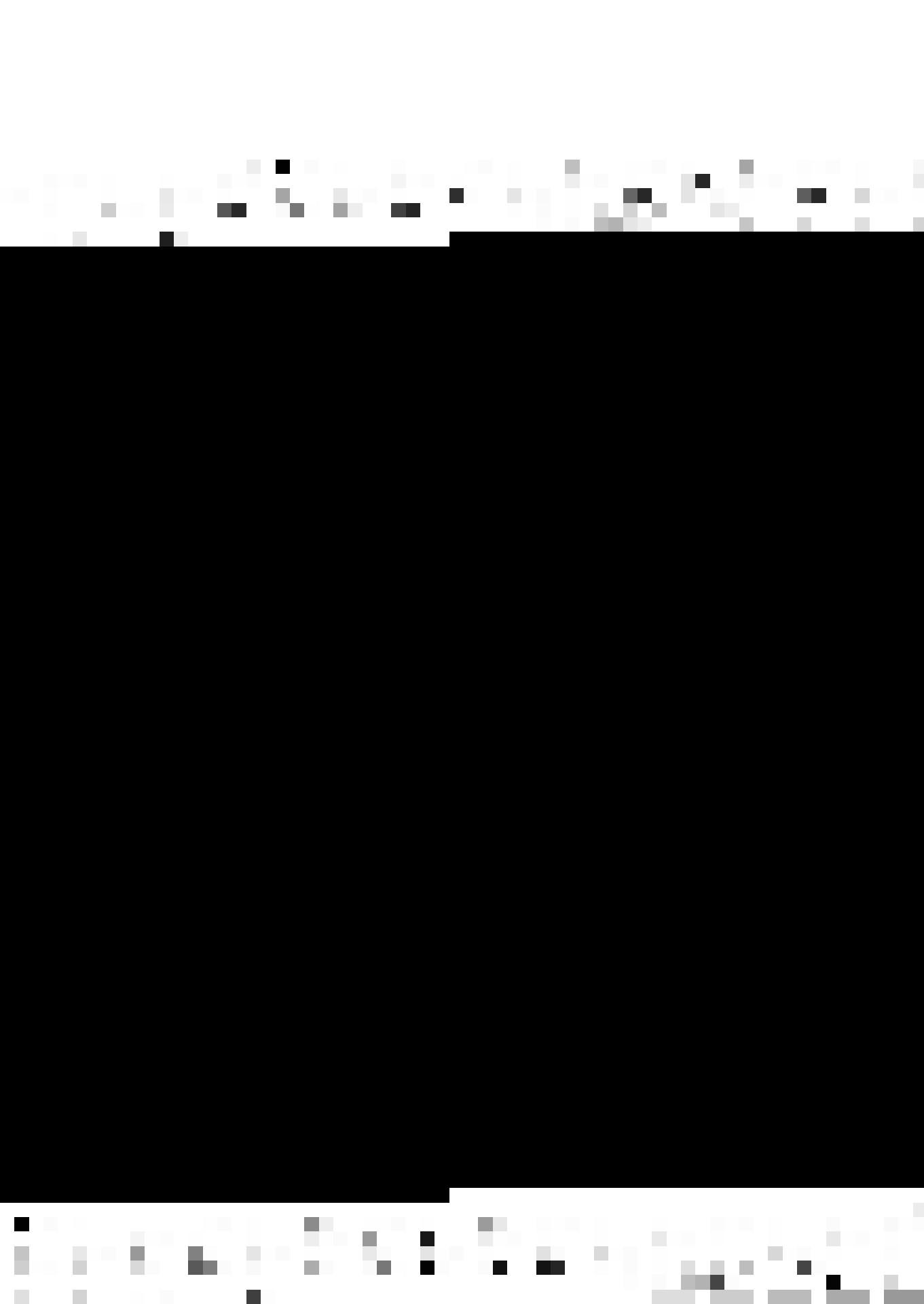
$$\langle q_i, z_i, q_{i+1}, z_{i+1}, \pi_1(i), \dots, \pi_n(i) \rangle.$$

(These productions express ways in which an entry can be erased from the pushdown store after its registers are filled in some order).

3. If  $\lambda(q, a, z)$  contains  $(r, Y)$ , then  $P$  contains

$$\langle q, z, p, X, i_1, \dots, i_n \rangle$$

$$\begin{aligned} \rightarrow (a \langle r, Y, p, X, i_1, \dots, i_n \rangle, \langle r, Y, p, X, \\ i_1, \dots, i_n \rangle, \dots, \langle r, Y, p, X, \\ i_1, \dots, i_n \rangle), \end{aligned}$$



and

$$\langle p, X, r, Y, i_1, \dots, i_n \rangle$$

$$\rightarrow (\langle p, X, q, Z, i_1, \dots, i_n \rangle a, \langle p, X, q, Z, i_1, \dots, i_n \rangle,$$

$$\dots, \langle p, X, q, Z, i_1, \dots, i_n \rangle)$$

for each  $p, q, r$  in  $Q$ ,  $X, Y, Z$  in  $\Gamma$ ,  $a$  in  $\Sigma \cup \{\epsilon\}$  and integers  $i_1, \dots, i_n$ .

(These productions represent moves which do not involve registers or alteration of the number of symbols in the pushdown store).

4. If  $\lambda(q, a, Z)$  contains  $(p, XY)$  and  $\lambda(r, b, Y)$  contains  $(s, i_1, \dots, i_n)$ , then  $P$  contains the productions:

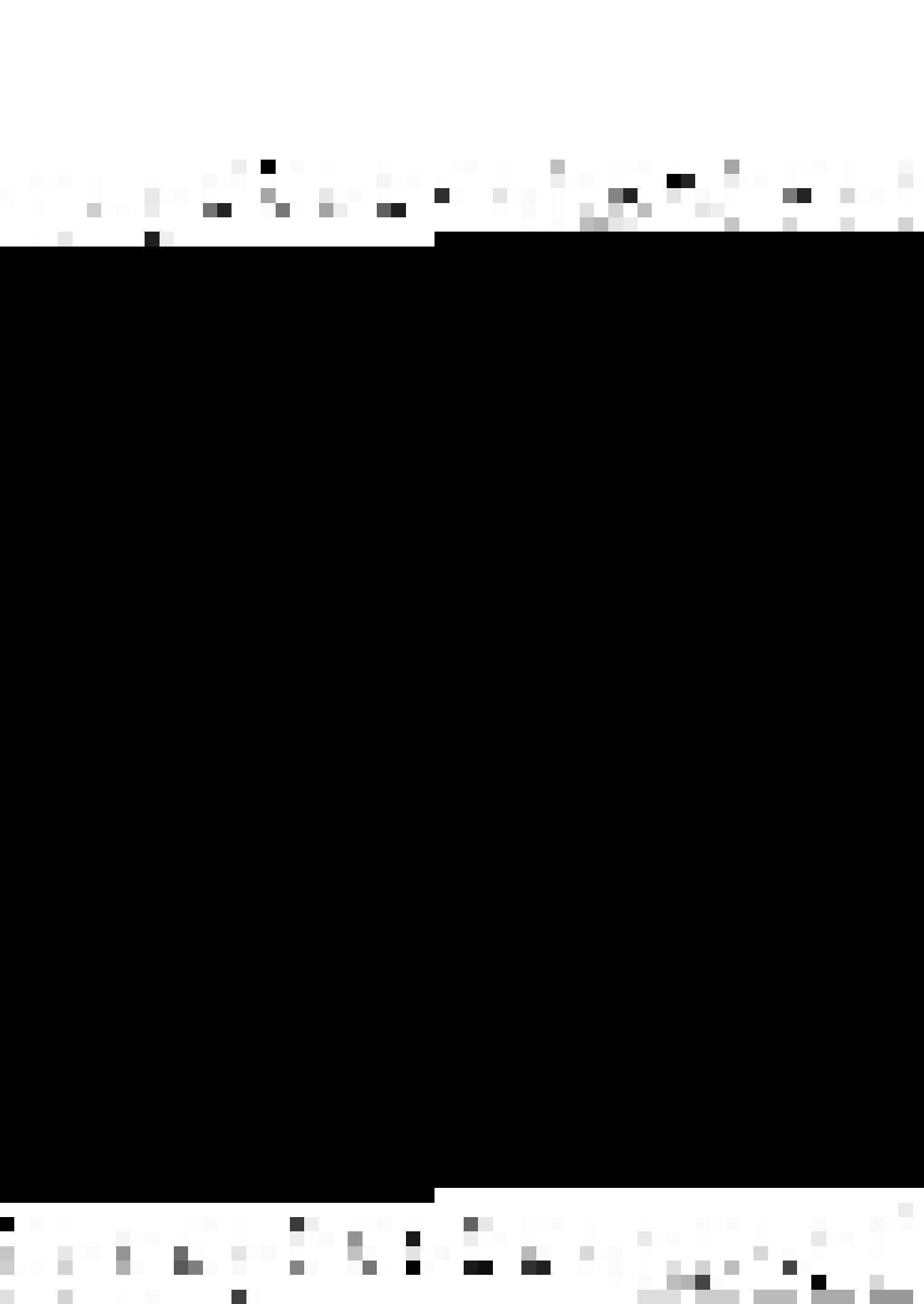
$$\langle q, Z, s, Y, i_1, \dots, i_n \rangle$$

$$\rightarrow (a[p, X, r]b, [p, X, r], \dots, [q, X, r]),$$

for all  $p, q, r, s$  in  $Q$ ;  $X, Y, Z$  in  $\Gamma$ ,  $a$  and  $b$  in  $\Sigma \cup \{\epsilon\}$  and integers  $i_1, \dots, i_n$ .

(These productions represent the situation in which  $M$  increases the number of symbols in the pushdown store, and when the new level is erased, transfers the contents of the  $j$ -th waiting register,  $1 \leq j \leq n$ , to the  $i_j$ -th register of the  $j$ -th set of registers, of the new top level).

5. If  $\mu(q, a, Z)$  contains  $(p, (w_1, i_1), \dots, (w_n, i_n))$ , then  $P$  contains:



$\langle q, z, p, z, i_1, \dots, i_n \rangle \rightarrow (a, w_1, \dots, w_n)$ ,  
 for any  $p$  and  $q$  in  $Q$ ,  $a$  in  $\Sigma \cup \{\epsilon\}$ ,  $w_j$ ,  $1 \leq j \leq n$ ,  
 in  $\Delta^*$  and integers  $i_1, \dots, i_n$ .

It is easy to prove by induction the following  
 (the proof will be omitted):

$$(a) ([q, z, p])^{n+1} \xrightarrow[G]{*} (x, y_1, \dots, y_n)$$

iff

$(q, x, z(t_0)^n, T_0) \vdash_{\bar{M}}^*(p, \epsilon, \emptyset, (y_1, \dots, y_n))$ , and

$$(b) (\langle q, z, p, y, i_1, \dots, i_n \rangle)^{n+1} \xrightarrow[G]{*} (x, y_1, \dots, y_n)$$

iff

$$(q, x, z(t_1, \dots, t_n), T_0)$$

$$\vdash_{\bar{M}}^* (p, \epsilon, Y(t'_1, \dots, t'_n), (y_1, \dots, y_n)),$$

where  $x$  is in  $\Sigma^*$ ,  $y_j$ ,  $1 \leq j \leq n$ , is in  $\Delta^*$ ,  $t_0 = (\emptyset)^k$ ,  
 $T_0 = (\emptyset)^n$ , the  $i_j$ -th component, of  $t_j$ ,  $1 \leq j \leq n$ , is  $\emptyset$   
 and  $t'_j$  is  $t_j$  with  $y_j$  in the  $j$ -th register of the set of  
 registers represented by  $t_j$ .

Now, from production (1) and statement (a) above,  
 it follows that  $T(\bar{M}) = T_1(G)$ .

Q.E.D.



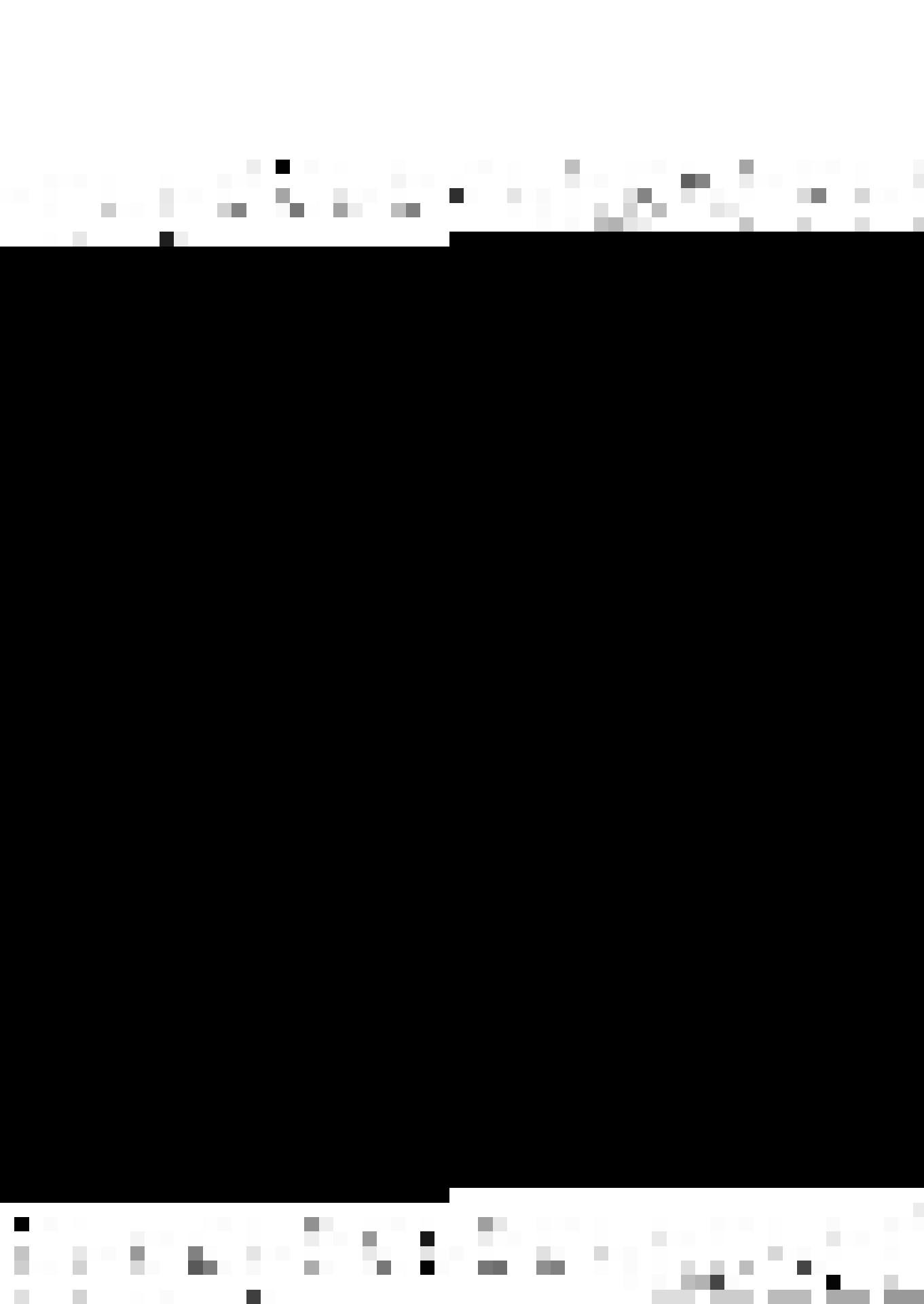
## APPENDIX H

In this appendix, we shall give an algorithm which tests whether the language generated by a given nCG  $G = (V_N, V_T, P, (S)^n)$ , contains  $\epsilon$ .

### Algorithm:

We construct sets  $N_0, N_1, \dots$  recursively as follows:

1. Let  $N_0 = \{A | A \rightarrow (\epsilon)^n, \text{ is in } P\}$ . Set  $i = 1$ .
2. Let  $N_i = \left\{ A | A \rightarrow (\alpha_1, (\alpha_2, \bar{\alpha}_2), \dots, (\alpha_n, \bar{\alpha}_n)), \right.$   
 $\text{is in } P, \text{ where } \alpha_j, 1 \leq j \leq n, \text{ is in } (N_{i-1})^*\} \cup N_{i-1}$ .
3. If  $N_i \neq N_{i-1}$ , then set  $i = i + 1$ , and go to step 2. Otherwise, let  $N_\epsilon = N_i$ .
4. If  $S$  is in  $N_\epsilon$ , then  $\epsilon$  is in  $L(G)$ . Otherwise,  $\epsilon$  is not in  $L(G)$ .



## APPENDIX I

In this appendix, we shall prove that the Parikh mapping of an n-coupled language is semilinear. Before proving the result (Lemma 1), we shall give a few definitions including that of the Parikh mapping (Definition 2).

### Definition 1:

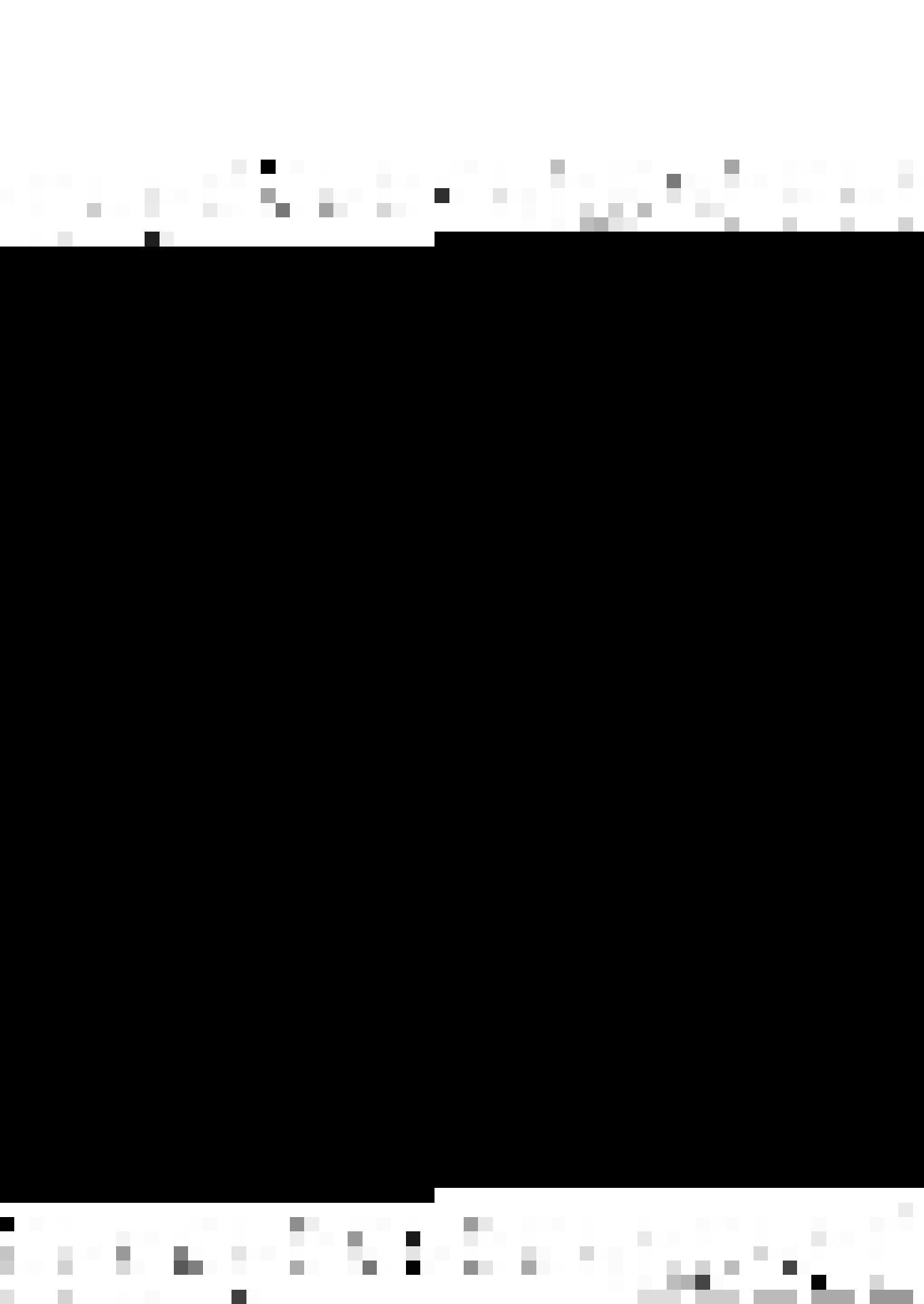
Let  $\mathbb{N}$  denote the non-negative integers and  $\mathbb{N}^m$  the cartesian product of  $\mathbb{N}$  with itself  $m$  times. If  $x = (x_1, \dots, x_m)$ , and  $y = (y_1, \dots, y_m)$ , are in  $\mathbb{N}^m$ , then  $x+y = (x_1+y_1, \dots, x_m+y_m)$ , and  $c(x_1, \dots, x_m) = (cx_1, \dots, cx_m)$ , where  $c$  is in  $\mathbb{N}$ . A subset  $Q$  of  $\mathbb{N}^m$  is said to be linear iff there is an integer  $k \geq 0$  and members  $v_0, \dots, v_k$  of  $\mathbb{N}^m$  such that  $Q = \{x | x = v_0 + n_1 v_1 + \dots + n_k v_k\}$ , where the  $n_i$ 's are non-negative integers.  $Q$  is said to be semilinear iff  $Q$  is the union of a finite number of linear sets.

### Definition 2:

For an alphabet  $V = \{a_1, \dots, a_m\}$ , the mapping  $\psi$  of  $V^*$  into  $\mathbb{N}^m$ , defined by,

$\psi(z) = (\#_{a_1}(z), \dots, \#_{a_m}(z))$ , where  $z$  is in  $V^*$  and  $\#_{a_i}(z)$  is the number of occurrences of  $a_i$  in  $z$ , is called the Parikh mapping. For a language  $L$ , the Parikh mapping is

$$\psi(L) = \{\psi(z) | z \text{ is in } L\}.$$



Lemma 1:

For any n-coupled language  $L$ , the set  $\psi(L)$  i.e. the Parikh mapping of  $L$ , is semilinear.

Proof:

Let  $G = (V_N, V_T, P, (S)^n)$ , be an nCG in normal form such that  $L = L(G)$ .

Let  $V'_N$  be a subset of  $V_N - \{S\}$  and  $L'$  a subset of  $L$  consisting of all those sentences whose derivations involve nonterminals which precisely comprise the set  $V'_N \cup \{S\} = U_0$ . Obviously, it is enough to show that  $\psi(L')$  is semilinear since  $L$  is a finite union of such  $L'$ .

Assume that the set  $U_0$  contains  $s$  elements. In what follows, we investigate the n-derivation trees in  $G$ .

Let  $H$  be the set consisting of strings  $z$  in  $V_T^*$  which are results of n-derivation trees in  $G$  such that the following conditions are satisfied:

(i) The n-nodes are labeled by elements of  $U_0$  and each element of  $U_0$  occurs atleast once as a label; and (ii) there is no n-path containing more than  $s+1$  n-nodes with the same label  $X$ . Clearly,  $H$  is a finite subset of  $L'$ . We define,

$$H_1 = \{\psi(x) \mid x \text{ is in } H\}.$$

Again,<sup>1</sup> for each  $Y$  in  $U_0$ , let

- 
1. Notice that because of the way we have defined an n-derivation tree in  $G$  (see section 1.3), the label of the n-root of an n-derivation tree  $T$  in  $G$  cannot be different from  $S$ , and the label of every pendant node in  $T$  has to be in  $V_T \cup \{\epsilon\}$ . This is the reason why  ${}^1T^Y$  will be defined in a somewhat awkward manner.



${}^1T^Y = \langle {}^1T_1^Y, \dots, {}^1T_n^Y \rangle$ , be an  $n$ -tuple of trees such that:  
 (i)  $T^Y = \langle T_1^Y, \dots, T_n^Y \rangle$ , is a subtree of an  $n$ -derivation tree in  $G$  rooted at an  $n$ -node labeled  $Y$ .  ${}^1T_i^Y$ ,  $1 \leq i \leq n$ , gets formed from  $T_i^Y$ , by deleting a subtree in  $T^Y$  rooted at an  $n$ -node labeled  $Y$ . (Thus, one pendant node of  ${}^1T_i^Y$ ,  $1 \leq i \leq n$ , has the label  $Y$  and all other pendant nodes have labels from  $V_T$ , and the label of the root of  ${}^1T_i^Y$  is  $Y$ ). (ii). There is no path in  ${}^1T_i^Y$ ,  $1 \leq i \leq n$ , containing more than  $s+1$  nodes with the same label  $X$ .

Let  $H_Y = \alpha_1 \dots \alpha_n$ , where  $\alpha_i$ ,  $1 \leq i \leq n$ , is the result of  ${}^1T_i^Y$ . Clearly, because of condition (ii)' as above,  $H_Y$  is a finite set. For  $\delta$  in  $H_Y$ , we denote by  $\gamma(\delta)$  the string obtained from  $\delta$  by erasing the occurrences of the symbol  $Y$ . We now define

$$H_2 = \{ \gamma(\gamma(\delta)) | \delta \text{ is in } H_Y \text{ for some } Y \text{ in } U \}.$$

Thus, both  $H_1$  and  $H_2$  are finite sets. Assume that

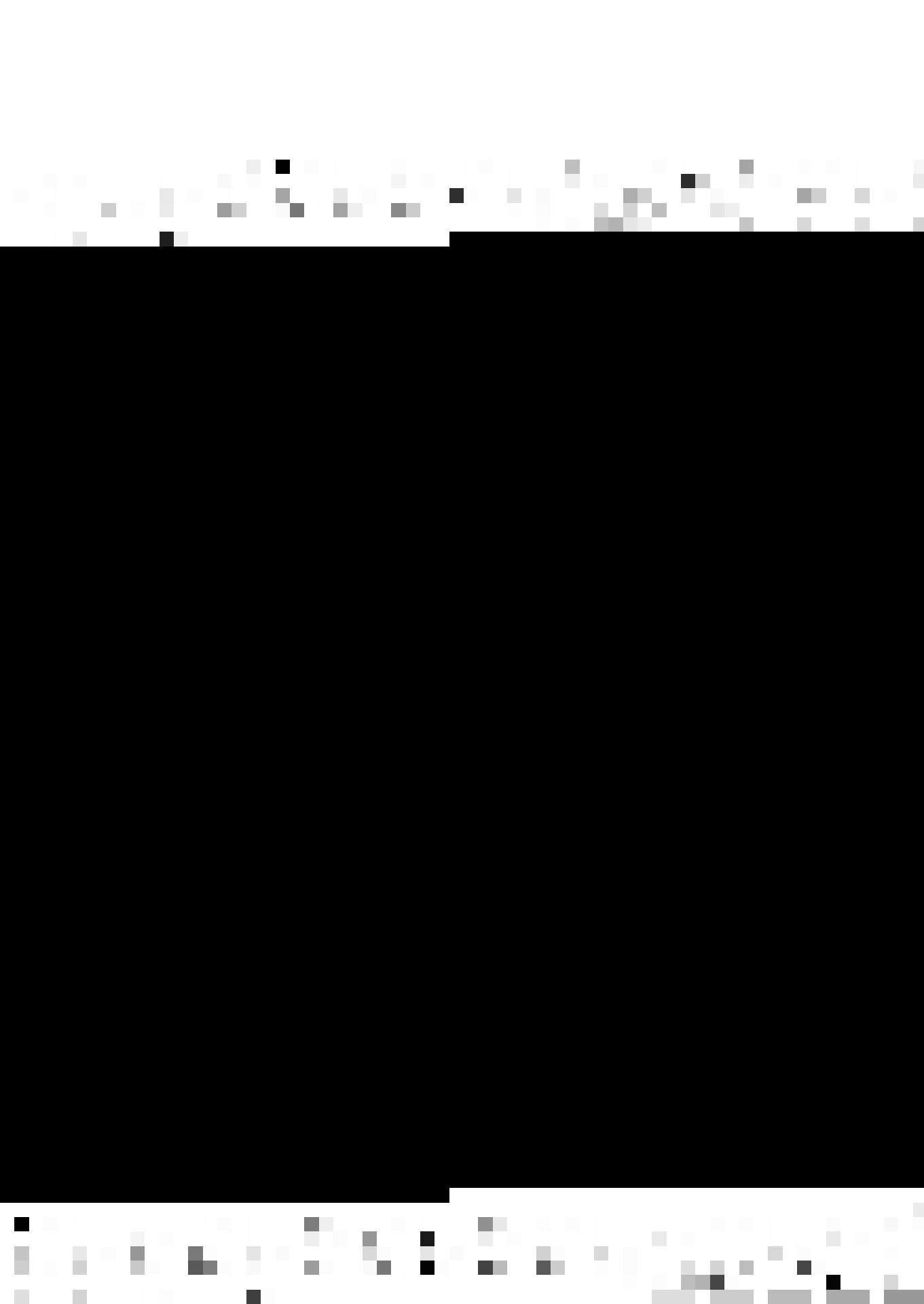
$$H_1 = \{ h_1, \dots, h_\alpha \}, \quad H_2 = \{ k_1, \dots, k_\beta \}.$$

Denote by  $K_i$ ,  $i = 1, \dots, \alpha$ , the set

$$\left\{ h_i + \sum_{j=1}^{\beta} x_j k_j \mid x_j \geq 0, \text{ for } j = 1, \dots, \beta \right\},$$

and let  $K$  be the union of the sets  $K_i$ .

1. Note that when a subtree rooted at an  $n$ -node  $M = \langle M_1, \dots, M_n \rangle$  detailed from an  $n$ -derivation tree  $T = \langle T_1, \dots, T_n \rangle$  in  $G$  so as to form an  $n$ -tuple of trees,  $T' = \langle T'_1, \dots, T'_n \rangle$ , the node  $M_i$ ,  $1 \leq i \leq n$  of  $T_i$  still remains in  $T'_i$ .
2.  $H_Y$  is the set of such  $H_Y$ .



By definition,  $K$  is semilinear. To prove Lemma 1, it suffices to show that

$$\psi(L') = K \quad (1)$$

We will prove that each side of (1) is included in the other.

Assume that  $v$  is in  $K$ . If  $v$  is one of the vectors  $h_i$ , then by the definition of the set  $H_1$ ,  $v$  is in  $\psi(H)$ , and consequently  $v$  is in  $\psi(L')$ . Proceeding inductively, we assume that

$$v_1 \text{ is in } K \cap \psi(L') \quad (2)$$

and claim that

$$v = v_1 + k_j, \text{ is in } \psi(L') \quad (3)$$

where  $j$ ,  $1 \leq j \leq \beta$ , is arbitrary. By the definition of  $H_2$ , there is a symbol  $Y$  in  $U_0$  and a string  $\delta$  in  $H_Y'$  such that  $k_j = \psi(\delta)$ . Since  $\delta$  is in  $H_Y'$ , there is an  $n$ -tuple of trees  ${}^1T^Y = \langle {}^1T_1^Y, \dots, {}^1T_n^Y \rangle$ , with properties specified above. Furthermore  $\delta = \alpha_1 \dots \alpha_n$ , where  $\alpha_i$ ,  $1 \leq i \leq n$ , is the result of  ${}^1T_i^Y$ ,  $Y$  occurs in each  $\alpha_i$  exactly once and  $\alpha_i$  contains no other nonterminals. The nonterminals occurring as node labels in each  ${}^1T_i^Y$ ,  $1 \leq i \leq n$ , belong to  $U_0$ . By (2), there is a sentence  $x$  in  $L'$  with  $v_1 = \psi(x)$  and a derivation  $(S)^n \xrightarrow[G]{*} (x_1, \dots, x_n)$ , such that  $x = x_1 \dots x_n$ , and the associated  $n$ -derivation tree  $T'$  has exactly the elements of  $U_0$  as labels of the nonpendant nodes of each tree comprising it. Hence, there is an  $n$ -node  $M = \langle M_1, \dots, M_n \rangle$  in  $T'$  labeled by  $Y$ . We 'imbed' the tree  ${}^1T_i^Y$ ,  $1 \leq i \leq n$ , in place of the node  $M_i$ . The resulting



$n$ -derivation tree<sup>1</sup>  $T'$  gives as a result (of  $T'$ ) a sentence  $x'$  in  $L'$  such that

$$\begin{aligned}\psi(x') &= \psi(x) + \psi(\gamma(\delta)) \\ &= v_1 + k_j = v\end{aligned}$$

Consequently, (3) holds true and we have completed the induction. Hence the right side of (1) is included in the left side.

Conversely, assume that  $v$  is in  $\psi(L')$ . Thus, there is a sentence  $x$  in  $L'$  such that  $v = \psi(x)$ . Let  $T = \langle T_1, \dots, T_n \rangle$  be the  $n$ -derivation tree of  $x$  in  $G$ . By the definition of  $L'$ , we assume that exactly the elements of  $U_0$  occur as labels of the  $n$ -nodes of  $T$ . If there is no  $n$ -path in  $T$  containing more than  $s+1$   $n$ -nodes with the same label, then  $x$  is in  $H$ . This implies that  $v$  is in  $H_1$  and hence  $v$  is in  $K$ .

Thus, we assume that there is an  $n$ -path in  $T$  containing at least  $s+2$   $n$ -nodes with the same label  $X$ . We choose  $s+2$   $n$ -nodes  $G_i = \langle G_{i1}, \dots, G_{in} \rangle$ ,  $0 \leq i \leq s+1$ , with the same label  $X$  in such a way that (i) for each  $i = 0, \dots, s$ ,  $G_{i+1}$  is in the subtree of  $T$  rooted at  $G_i$ , and (ii) there is no  $n$ -path containing

1. Remember (from section 1.3) that the only role of permutations associated with an  $n$ -node  $M = \langle M_1, \dots, M_n \rangle$ , of an  $n$ -derivation tree in  $G$  is to specify the correspondence between the nonpendant direct descendants of  $M_i$ ,  $1 \leq i \leq n$ .



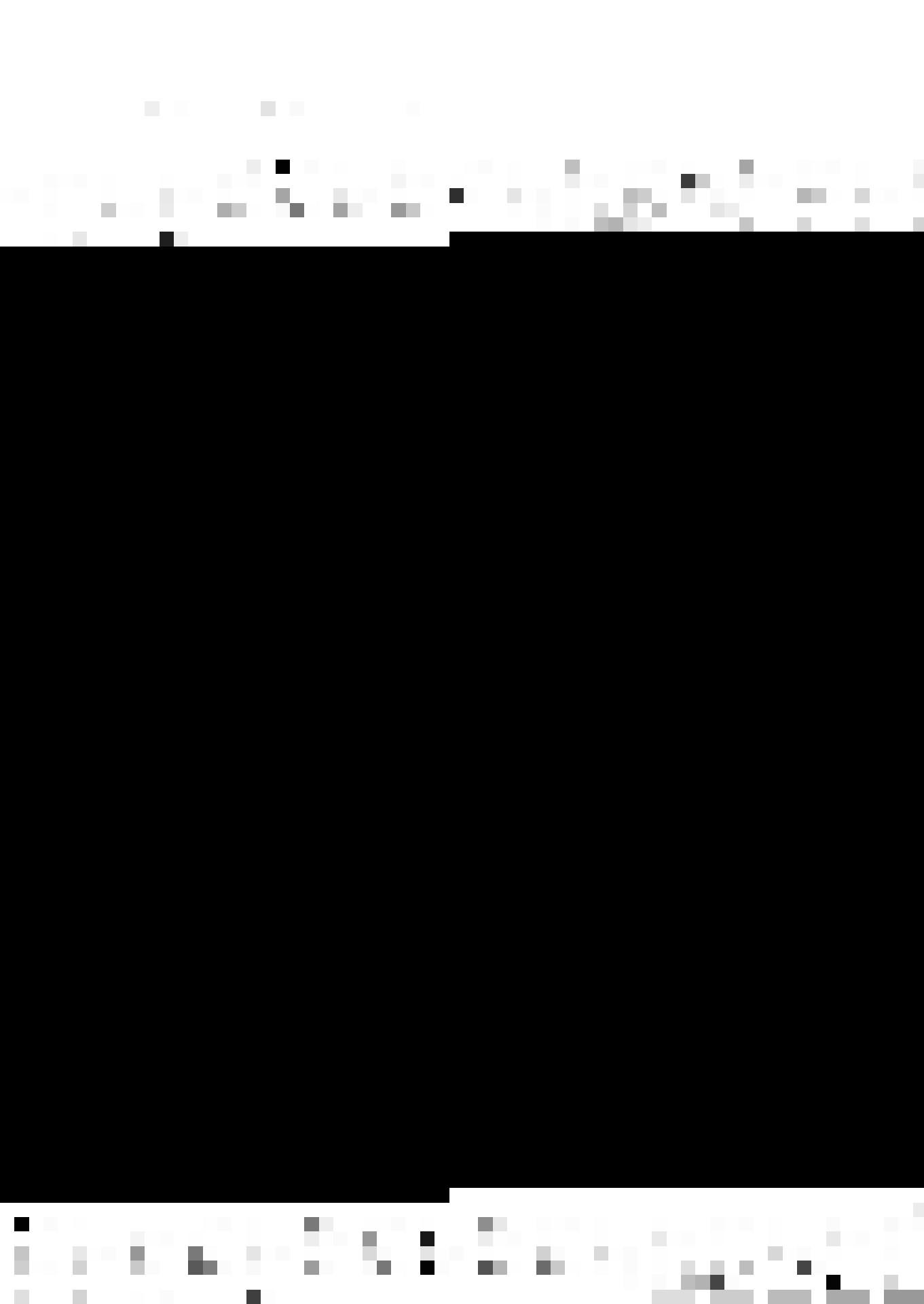
more than  $s+1$  n-nodes with the same label in the subtree  $T^{G_1}$  of  $T$  rooted at  $G_1$ . There is a number  $p$ ,  $1 \leq p \leq s$ , such that the same nonterminals occur as n-node labels in the subtrees  $T^{G_p}$  and  $T^{G_{p+1}}$  of  $T$ , rooted at the n-nodes  $G_p$  and  $G_{p+1}$ , respectively. This follows because there are atmost  $s$  nonterminals as n-node labels in  $T^{G_1}$ , and atleast one nonterminal as an n-node label in  $T^{G_{s+1}}$  (the subtree of  $T$  rooted at the n-node  $G_{s+1}$ ). We now form a new n-derivation tree  ${}^2T$  in  $G$  by replacing in  $T$  the subtree rooted at  $G_p$  by that rooted at  $G_{p+1}$ . Denote by  ${}^1T' = \langle {}^1T'_1, \dots, {}^1T'_n \rangle$ , the ordered set of trees formed from  $T^{G_p}$  as follows: Let  $T^{G_p} = \langle T_1^{G_p}, \dots, T_n^{G_p} \rangle$ , and  $T^{G_{p+1}} = \langle T_1^{G_{p+1}}, \dots, T_n^{G_{p+1}} \rangle$ .  ${}^1T'_i$ ,  $1 \leq i \leq n$ , is formed from  $T_i^{G_p}$  by replacing the tree  $T_i^{G_{p+1}}$  by the node  $G_{(p+1)i}$ . The string  $\beta_1 \dots \beta_n$ , where  $\beta_i$ ,  $1 \leq i \leq n$ , is the result of  ${}^1T'_i$  belongs to set  $H_x$  (by the definition of this set). Consequently, if  $x'$  is the result of  ${}^2T$ , we obtain,

$$v = \psi(x) = \psi(x') + k_j, \quad (4)$$

for some  $j$ ,  $1 \leq j \leq \beta$ .

Clearly,  ${}^2T$  has fewer n-nodes than  $T$ . If there is an n-path in  ${}^2T$  containing more than  $s+1$  n-nodes with the same label, we repeat the procedure above for  ${}^2T$  and obtain an n-derivation tree  ${}^3T$  whose result is  $x''$ . The equation corresponding to (4) is now

$$\psi(x') = \psi(x'') + k_j,$$



where  $1 \leq j' \leq \beta$ . Continuing in this fashion, we finally obtain an  $n$ -derivation tree where there is no  $n$ -path containing more than  $s+1$   $n$ -nodes with the same label. Thus we can write  $v$  in the form

$$v = h_i + \sum_{j=1}^{\beta} x_j k_j, \quad 1 \leq i \leq \alpha; \quad x_j \geq 0 \text{ for } 1 \leq j \leq \beta.$$

But this means that  $v$  is in  $K$ , and consequently the left side of (1) is included in the right side.

Q.E.D.



## APPENDIX J

In this appendix, we shall show that the class of equal matrix languages of order  $n$  is the same as the class of  $n$ -right linear coupled languages (Theorem 1) by proving Lemma 1 and Lemma 2. The proofs of Lemmas 1 and 2 are similar to Lemma 5.2.2 and Lemma 5.2.3, respectively, and so we shall only give the necessary constructions in the proofs.

Lemma 1:

$$\mathcal{R}_n^L \subset \Sigma_n$$

Proof:

Let  $G = (V_N, V_T, P, (S)^n)$  be an nRLCG such that  $S$  is not on the right hand side of any production<sup>1</sup>, where  $V_N = \{A_{11}, \dots, A_{1k}\}$ . We construct an EMG  $G'$  of order  $n$ , such that  $L(G) = L(G')$ , as follows:

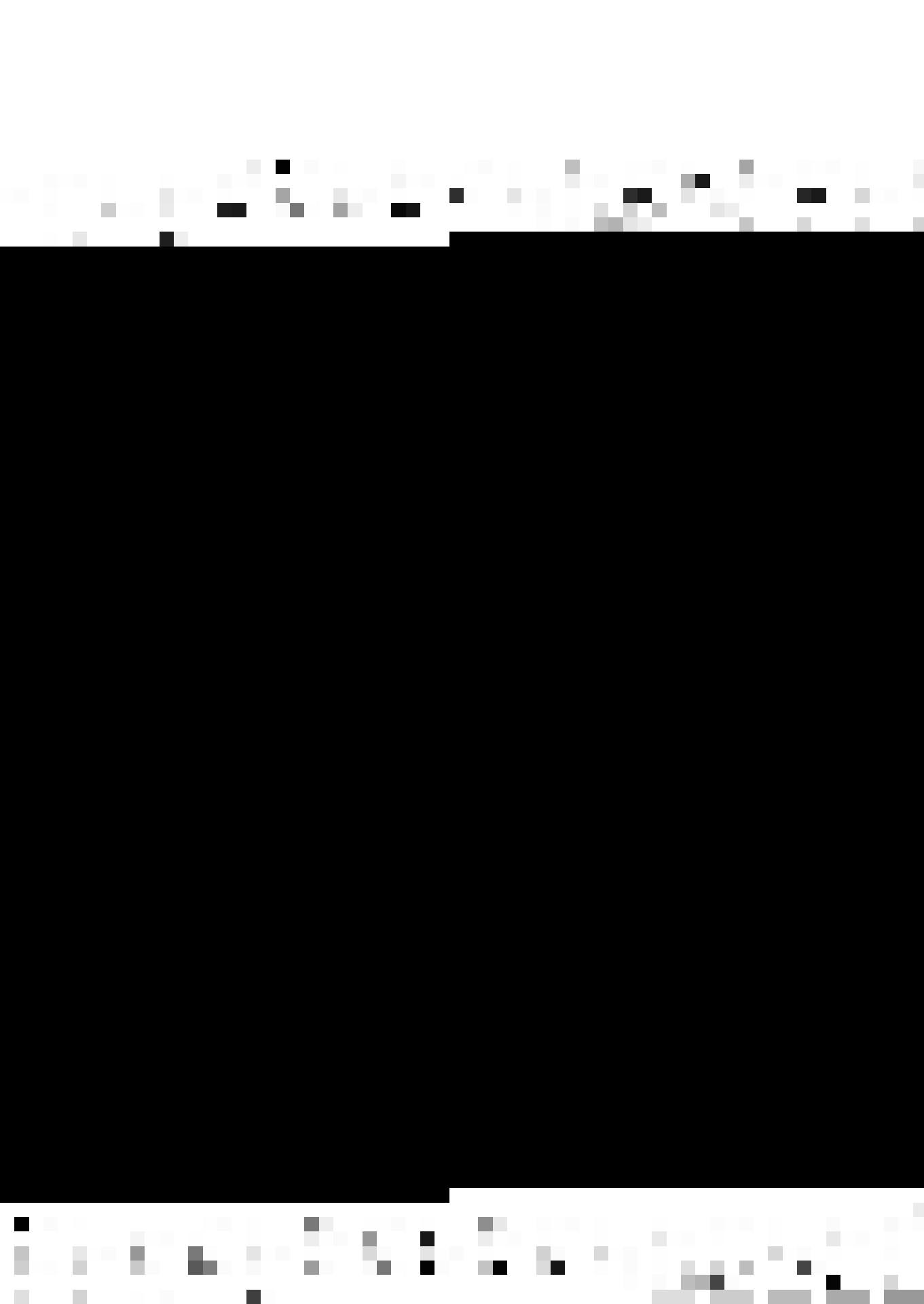
$$G' = (V', V_T, P', S), \text{ where}$$

$$V' = V_T \cup \left\{ \langle B_1, \dots, B_n \rangle \mid B_i, 1 \leq i \leq n, \right. \\ \left. \text{is in } \{A_i, A_{i1}, \dots, A_{ik}\} \right\},$$

and the rules of  $P'$  are constructed as follows:

(a) (i)  $[S \rightarrow A_1 \dots A_n]$ , is in  $P'$ .

1. It is easy to see that given an nRLCG  $G'$ , we can construct an nRLCG  $G$  such that  $L(G) = L(G')$  and  $G$  satisfies this constraint.



(ii) For each production:

$S \rightarrow (x_1 A_{1p}, \dots, x_n A_{1p})$ , in  $P$ ,  $1 \leq p \leq k$ ,  
 $x_1, \dots, x_n$  in  $V_T^*$ , we have in  $P'$  the rule:

$$\boxed{\begin{array}{l} A_1 \rightarrow x_1 A_{1p} \\ \vdots \\ A_n \rightarrow x_n A_{np} \end{array}}$$

(b) For each production:

$A_{1x} \rightarrow (x_1 A_{1p}, \dots, x_n A_{1p})$ , in  $P$ ,  
 $1 \leq x, p \leq k$ ,  $x_1, \dots, x_n$  in  $V_T^*$ , we have in  $P'$  the rule:

$$\boxed{\begin{array}{l} A_{1x} \rightarrow x_1 A_{1p} \\ \vdots \\ A_{nx} \rightarrow x_n A_{np} \end{array}}$$

(c) For each production:

$A_{1x} \rightarrow (w_1, \dots, w_n)$  in  $P$ ,

$1 \leq x \leq k$ ,  $w_1, \dots, w_n$  in  $V_T^*$ , we have in  $P'$  the rule:

$$\boxed{\begin{array}{l} A_{1x} \rightarrow w_1 \\ \vdots \\ A_{nx} \rightarrow w_n \end{array}}$$

(d) For each production:

$S \rightarrow (w_1, \dots, w_n)$ , in  $P$ ,  $w_1, \dots, w_n$  in  $V_T^*$ , we have in  $P'$  the rule:



$$\boxed{\begin{array}{l} A_1 \rightarrow w_1 \\ \vdots \\ \vdots \\ A_n \rightarrow w_n \end{array}}$$

Q.E.D.

Lemma 2:

$$\mathcal{E}_n \subset \mathcal{R}_n^L$$

Proof:

Let  $G = (V, I, P, S)$  be an EMG of order  $n$ . We construct an nRLCG  $G'$  such that  $L(G) = L(G')$ , as follows:

$G' = (V'_N, I, P', (S)^n)$ , where  $V'_N = (V - I) \cup \{s\}$ , and  $P'$  is constructed as follows:

(a) For each rule:

$[S \rightarrow f_1 A_1 \dots f_n A_n]$  in  $P$ ,  $f_1, \dots, f_n$  in  $I^*$ ,  $\langle A_1, \dots, A_n \rangle$  in  $V - I$ , we have in  $P'$  the production:

$$S \rightarrow (f_1 \langle A_1, \dots, A_n \rangle, \dots, f_n \langle A_1, \dots, A_n \rangle).$$

(b) For each rule:

$$\boxed{\begin{array}{l} A_{11} \rightarrow f_1 B_1 \\ \vdots \\ \vdots \\ A_n \rightarrow f_n B_n \end{array}} \quad \text{in } P, \text{ where}$$

$f_1, \dots, f_n$  are in  $I^*$ ,  $\langle A_1, \dots, A_n \rangle, \langle B_1, \dots, B_n \rangle$  are in  $V - I$ , we have in  $P'$  the production:

$$\langle A_1, \dots, A_n \rangle \rightarrow (f_1 \langle B_1, \dots, B_n \rangle, \dots, f_n \langle B_1, \dots, B_n \rangle).$$



(c) For each terminal rule of the form:

$$\begin{bmatrix} A_1 \rightarrow f_1 \\ \vdots \\ \vdots \\ A_n \rightarrow f_n \end{bmatrix} \quad \text{in } P, \text{ where}$$

$\langle A_1, \dots, A_n \rangle$  is in  $V-I$ , and  $f_1, \dots, f_n$  are in  $I^*$ , we have in  $P'$  the production:

$$\langle A_1, \dots, A_n \rangle \rightarrow (f_1, \dots, f_n)$$

Q.E.D.

From Lemmas 1 and 2, we have:

Theorem 1:

$$\mathcal{E}_n = \mathcal{R}_n^L$$



45490

EE-1775-D-VAI-CL

